White Papers and Patents – Palash Sethi

Table of Contents

Note – Please go through White Papers of respective patents to get fundamental understanding of the idea. Patent Documents contain legalese along with fundamental ideas.

BUSINESS:	**SITE:**	
DATE OF SUBMISSION:		
DETAILS OF INVENTORS		
Lead Inventor	Name :	Palash Sethi
	Lead / Co-Inventor :	Lead
	Father's / Husband's Name :	
	Address (Present) :	
	Address (Permanent) :	
	Nationality :	
	PAN No. :	
	Aadhaar No. :	
	Email ID (alternate) :	palash1995.18@gmail.com
	Percentage Contribution % :	50
	Digital Signature Image (JPEG Format):	
Co-Inventor – 2	Name :	Shailesh Kumar
	Lead / Co-Inventor :	Co-Inventor
	Father's / Husband's Name :	
	Address (Present) :	
	Address (Permanent) :	
	Nationality :	
	PAN No. :	
	Aadhaar No. :	
	Email ID (alternate) :	
	Percentage Contribution % :	50
	Digital Signature Image (JPEG Format):	

	Yes √	**No √**	**Details (If Yes, Provide Details)**
Is this invention out of external collaborative research?		√	
Is this invention disclosed to outside party?		√	
Do you plan to present the work in public near future?	√		
Do you have a lab notebook?		√	
Have this invention reduced to practice / prototype?		√	

TITLE OF INVENTION

Complexity boosted sampling via Oracle for efficiently training Neural Networks

ABSTRACT OF THE INVENTION (Max 300 Words)

Neural Networks need large amount of training data for learning non-linear, multi-variable complex system. In principle the Neural Networks can be trained to approximate system's behavior with large set of training data over the entire input range. Generating data for the entire input range is cumbersome and time consuming. Using Active complexity-boosted sampling methodology optimizes the time to train the ANNs and increase accuracy of trained network. An adaptive sampling algorithm is conceived to generate dense training dataset for input range and iterate the training of ANN further for the range with dense dataset for the areas of in low accuracy. The proposed complexity aware smart sampling can be used for training any learning algorithm.

BACKGROUND OF THE INVENTION

Non-Linear Complex simulations of physical systems using the laws of physics such as thermodynamics, chemical reactions, differential equations etc. are simulated to solve for a given input. These simulations are time-taking and are unfeasible to run multiple times for the entire operational range in business situations. Using Linear Approximation of the complex simulations helps in faster results generation with acceptable errors, it is less accurate but quick to do and works only far a small range of input domain. Neural Networks once trained enough can approximate these input-output mapping better than Linear Approximations and can give simulation results faster than traditional simulation algorithms.

Neural Networks can be trained in two ways:

1. Using pre-collected dataset
2. By sampling mathematical models which can simulate physical systems

For training Neural Networks to approximate a black-box model, the simulation dataset needs to be sampled using a black-box model, or via an Oracle function which can efficiently simulate the given physical system. This sampling needs to be optimized in such a way that more data is sampled where the complexity of the manifold is high, while keeping the number of total data points sampled at minimum. Curriculum Learning is a classic learning strategy which states that learning algorithms can be trained in an efficient manner by gradually exposing the algorithm to difficult or complex data points with iterations. We propose an iterative training approach which starts with a coarse level dataset and iteratively employs a curriculum learning strategy to sample highly complex data points.

In case of a given oracle, this training data is acquired by sampling the points in a given input domain and running this set of input data through the oracle to generate outputs. For a supervised learning paradigm, the sampling methodology applied for this process is uniform random sampling. Uniform random sampling assumes that all points in the input domain are equally significant for the learning task, which, is not sufficient to map highly complex input-output mapping.

Proposed sampling methodology implements a complexity-based sampling which trains the Neural Network in complex mapping regions, by iteratively sampling the oracle and training the neural network in complex regions.

DETAILED DESCRIPTION OF THE INVENTION

(a) Proposes a complexity-based sampling methodology to train Neural Network via an oracle function
(b) The algorithm samples points proportional to the complexity of a region in an iterative fashion, while training a neural network.
(c) For finding pure regions of near constant complexity, a regression tree is used. Same function can also be achieved by other methods that divide the space based on a criteria, such as KD Trees.
(d) Complexity is defined by the accuracy of the trained neural network in a given region during an iteration.
(e) Data generated is weighed in proportion to its iteration number while training the neural network, to avoid excessive overfitting.
(f) The network is trained until a threshold of accuracy is reached on the testing dataset.
(g) Proposed method performs Iterative Curriculum Sampling.

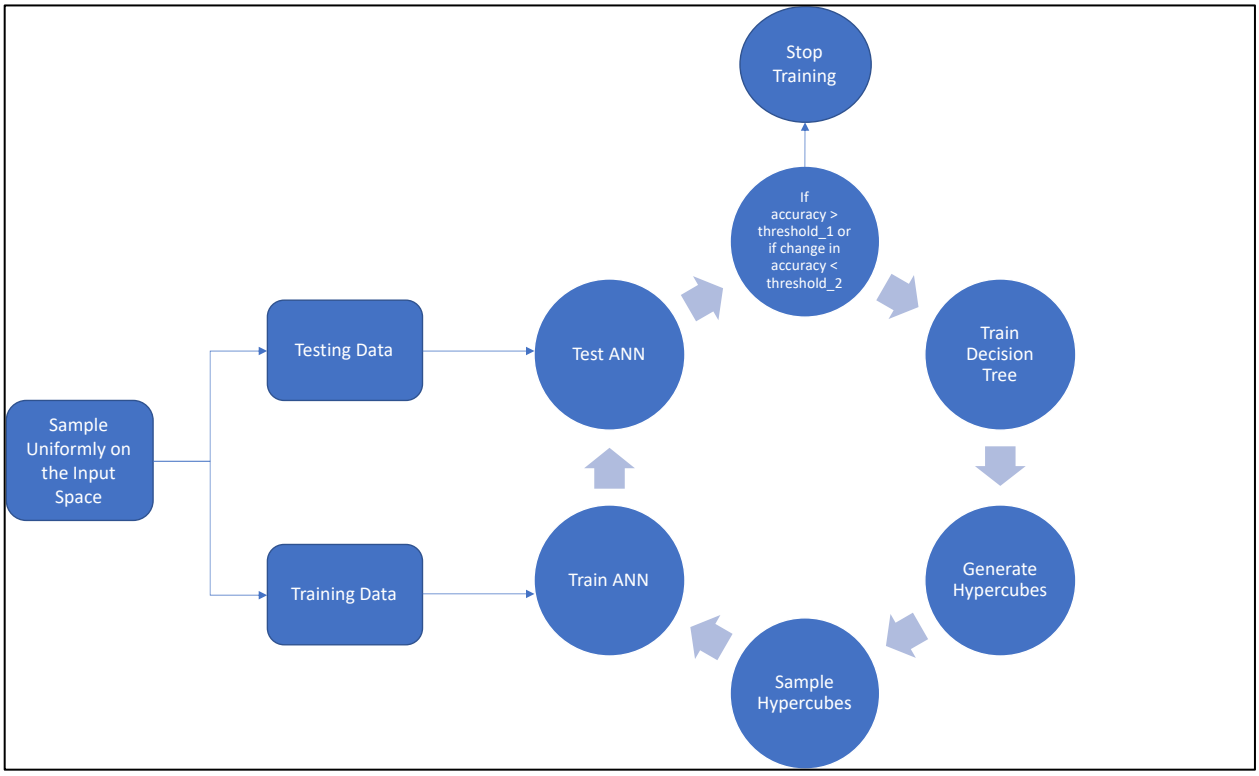


Figure 1Sampling Algorithm Stages

The methodology starts with an oracle function that is attempting to simulate a complex physical or chemical process with an objective function. Oracle function is often supported with domain specific tools, such as a simulation software. A multidimensional input range is selected based on domain knowledge.

Architecture of the neural network is fed in as a hyperparameter to the training pipeline.

Uniform random sampling is performed in the given input range, where the number of samples generated per iteration is a hyperparameter to the training pipeline. These input samples are fed into the oracle function in a parallelizable fashion and outputs for the sampled inputs are generated. This dataset is divided into training and test dataset. λ, the batch importance parameter assigned to this initial training set is 1.0.

The neural network with a given architecture is initialized with random weights and biases.

Training process begins by choosing an optimizer and its learning rate. We use Adam Optimizer. The neural network is trained on the sampled dataset for a given number of epochs. The loss function used to train this neural network is defined as:

$$\frac{1}{\max(i)}\sum_{i}\frac{1}{N}\sum_{j}\lambda^{(i-\max(i))} * \left(y_{ij}-\bar{y}_{ij}\right)^2$$

where i is the sampling iteration

j is the sample in i^{th} iteration

max (i) is the current sampling iteration

N is the number of samples generated per iteration

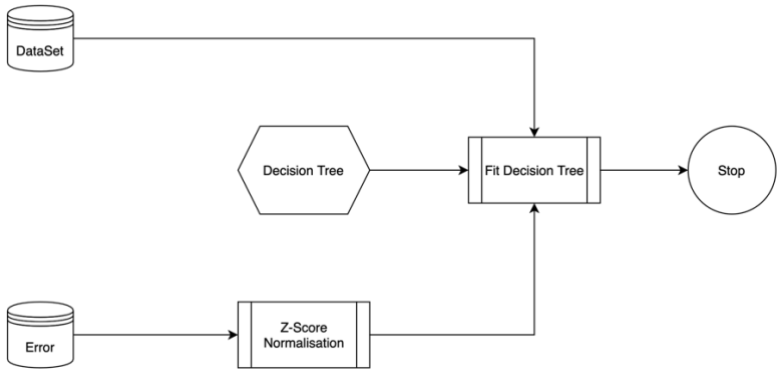
λ is the batch importance parameter

y_{ij} and $\bar{y}_{ij}$ are the target and predicted output values

Trained neural network is tested on the training data, and mean squared error is calculated for each sample in the training data.

Since no geometry could capture actual error cloud, we use n-dimensional hypercubes to approximate the error distribution in the space. We use decision trees to identify hypercubes.

A regression tree is trained on the sampled input data against the mean squared error of outputs predicted by neural network. This is done to identify pure regions in the error domain. These regions have almost constant error values. To identify these regions, a depth first search algorithm is implemented on the trained decision tree, and the decision rules leading to leaf nodes are identified. These decision rules identify pure n-dimensional hypercubes.

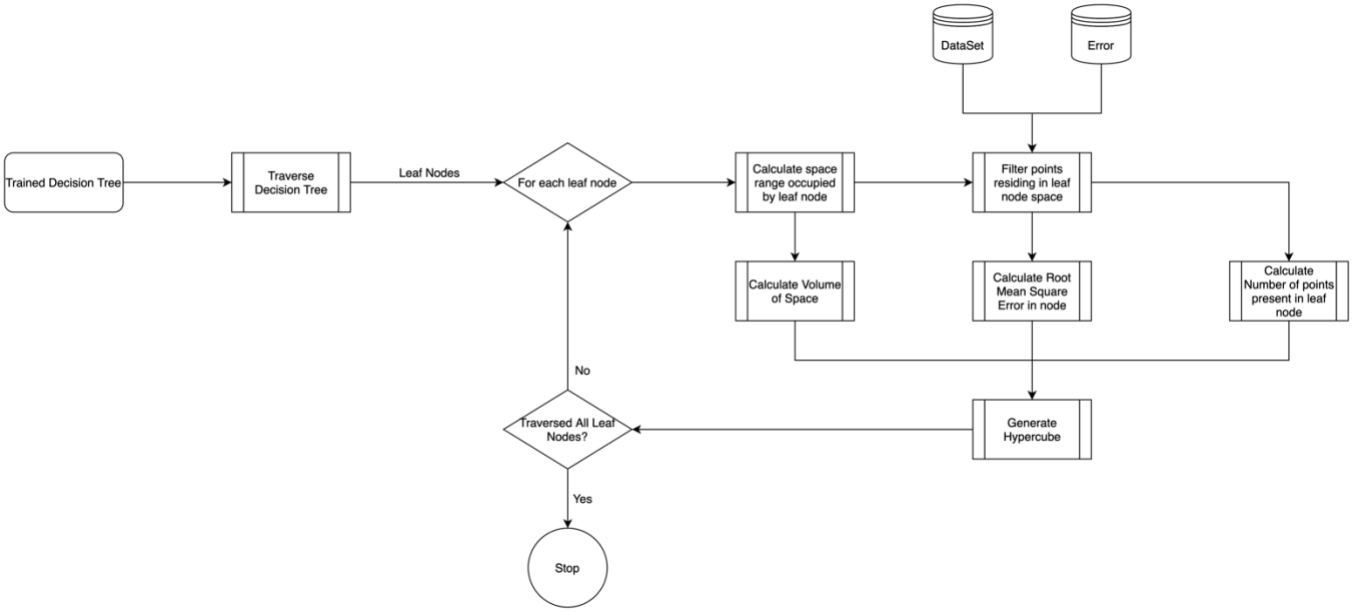


A hypercube is defined by the volume, the number of points and the average error value associated to these points in the encompassed region.

$$hypercube : \{n_k^t, e_k^t, v_k^t\}_k^L$$

where n_k^t, e_k^t, v_k^t are the number of points, their average error and the volume of k^{th} hypercube

L is the number of leaf nodes learnt by regression tree



A normalizing parameter, Z is calculated for the entire set of hypercubes

$$Z = \sum_k (e_k^t \times v_k^t)^\alpha$$

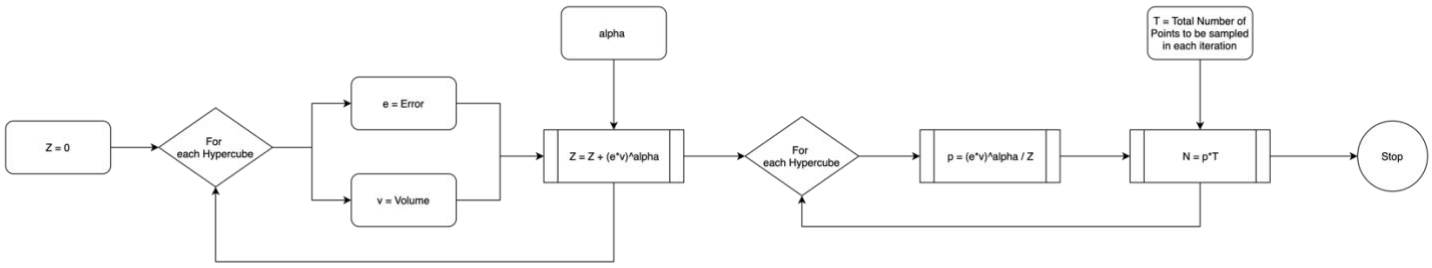
where α is a hyperparameter required for exponential sampling numbers

We define target density for each hypercube as

$$p_k^t = \frac{(e_k^t \times v_k^t)^\alpha}{Z}$$

For each hypercube, target density is calculated. New samples are uniformly sampled in each hypercube, with the number of samples being proportional to the target density of the hypercube, that is, spread and intensity of the error values in a pure region decides the number of points to be sampled in this iteration.

$$N_k^t = N \times p_k^t$$



The new samples are passed into the oracle function to generate outputs. The input and output points generated across the hypercubes are concatenated to create the second batch of training data. This training data is given a batch importance parameter proportional to the iteration number.

The training process is continued until the neural network achieves an acceptable error on the test dataset.

This methodology helps in adopting smart sampling of large input space with focused approach on areas of input space with low accuracy. Thereby capturing maximum non-linearity of a system and providing good representation in neural networks.

The benefits of the proposed training algorithm:

1. Amount of training data required is minimum
2. Accuracy of the model is higher
3. Mimics the human learning process

A use case scenario using "**Styblinski Tang**" objective function with visualization of accuracy loss in a sample space is shown below.

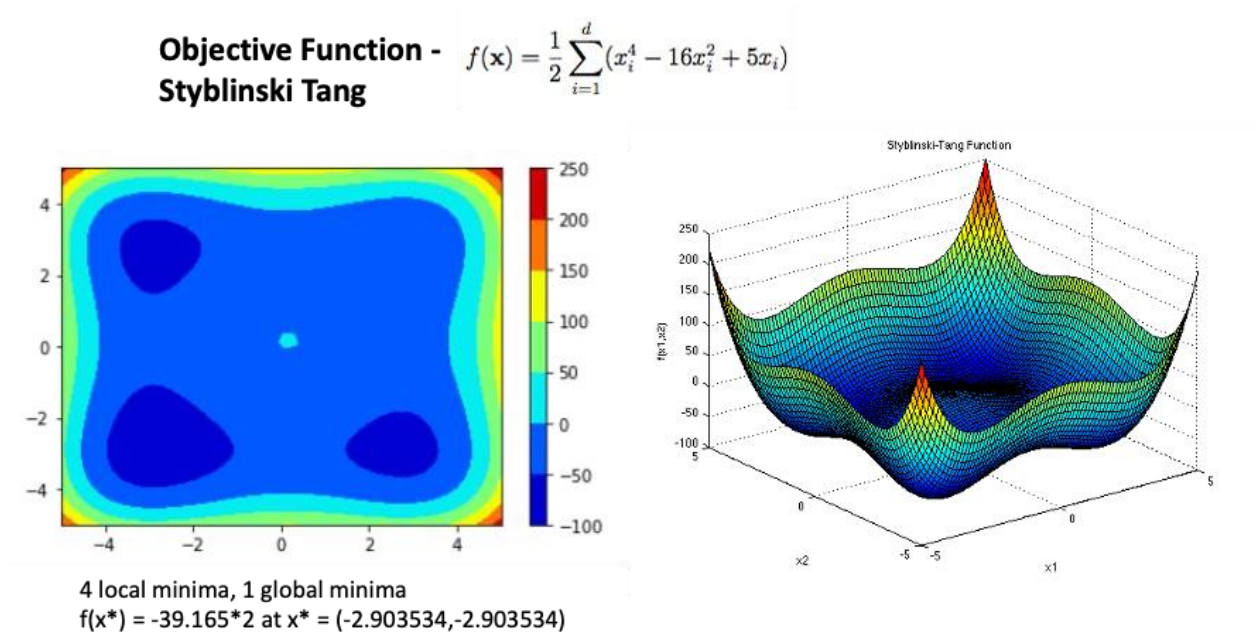
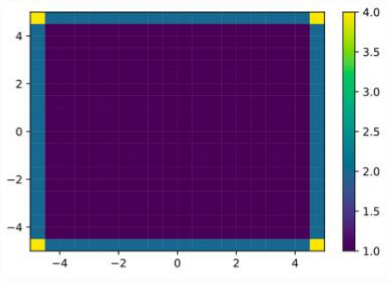
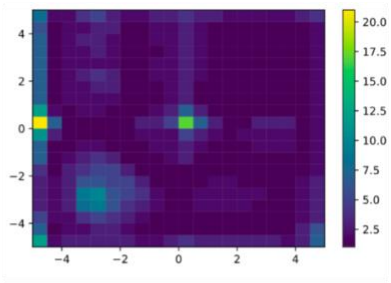


Figure 2 Oracle Visualization for the complete input space

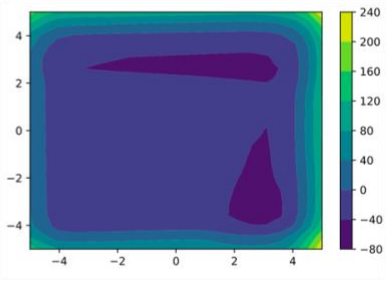
'Styblinski Tang' is a benchmark objective function used for testing optimization algorithms. It consists of 4 local minima and 1 global minima. Although the objective function can be used for multi-dimension input, we use a 2-D version of this objective function for visualizing various iterations of the training algorithm. As can be seen in Figure 2, the function is highly non-linear near the manifold of local minimas.

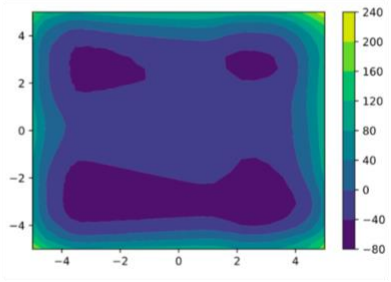
Number of points sampled



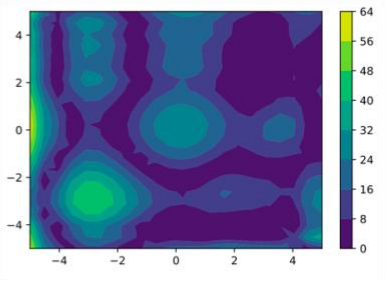


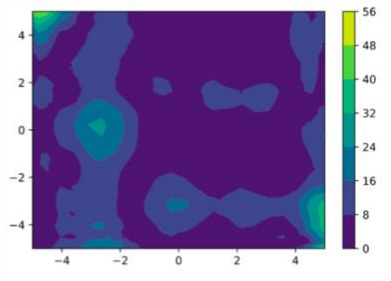
Predictions post training



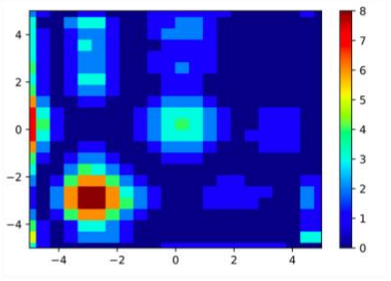


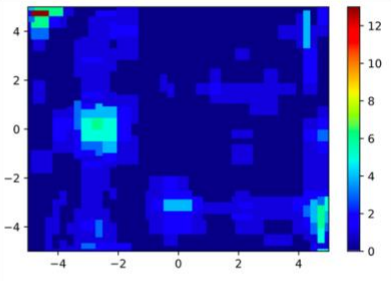
Prediction Error post training



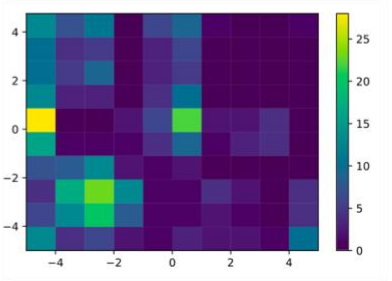


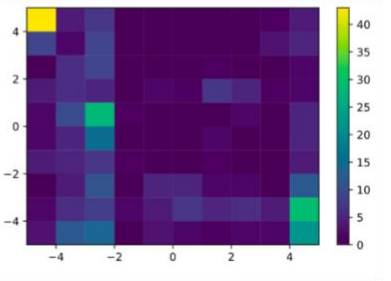
Generated Hypercubes



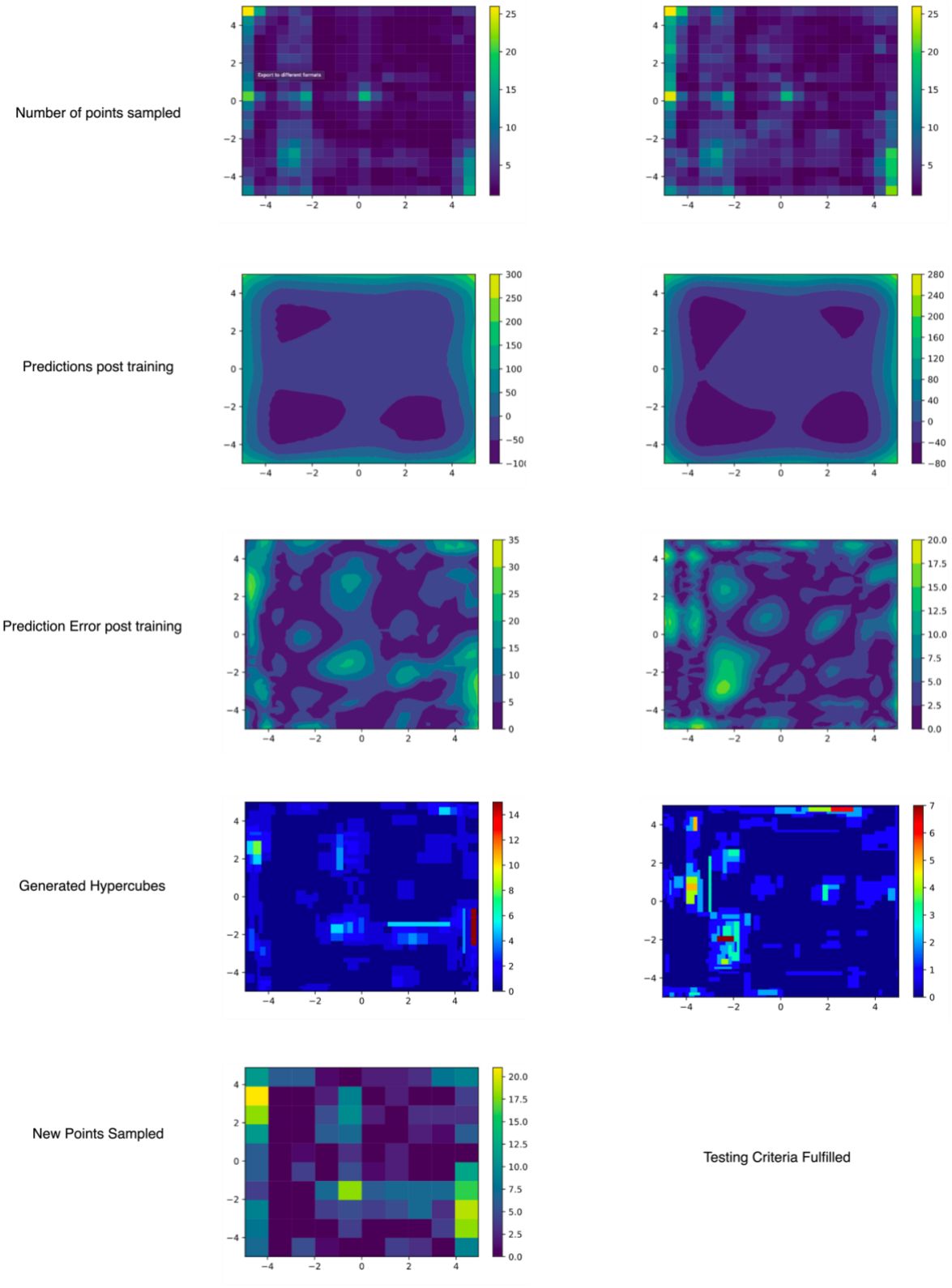


New Points Sampled





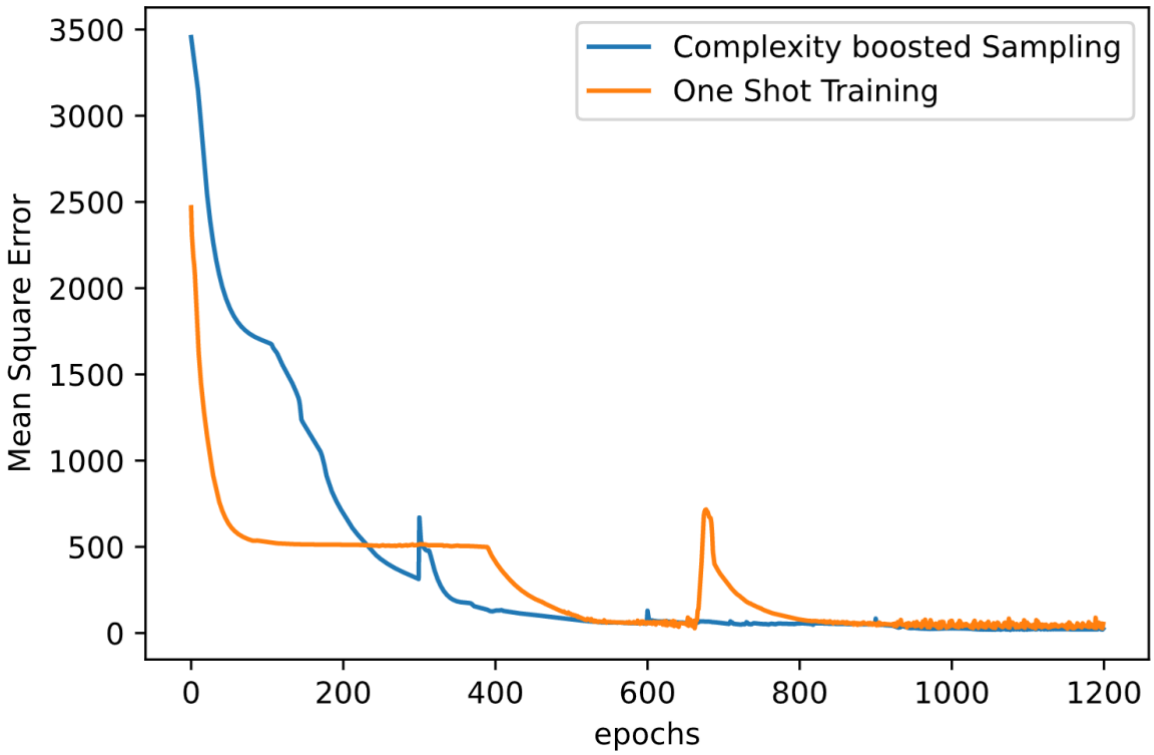
T = 1 T=2

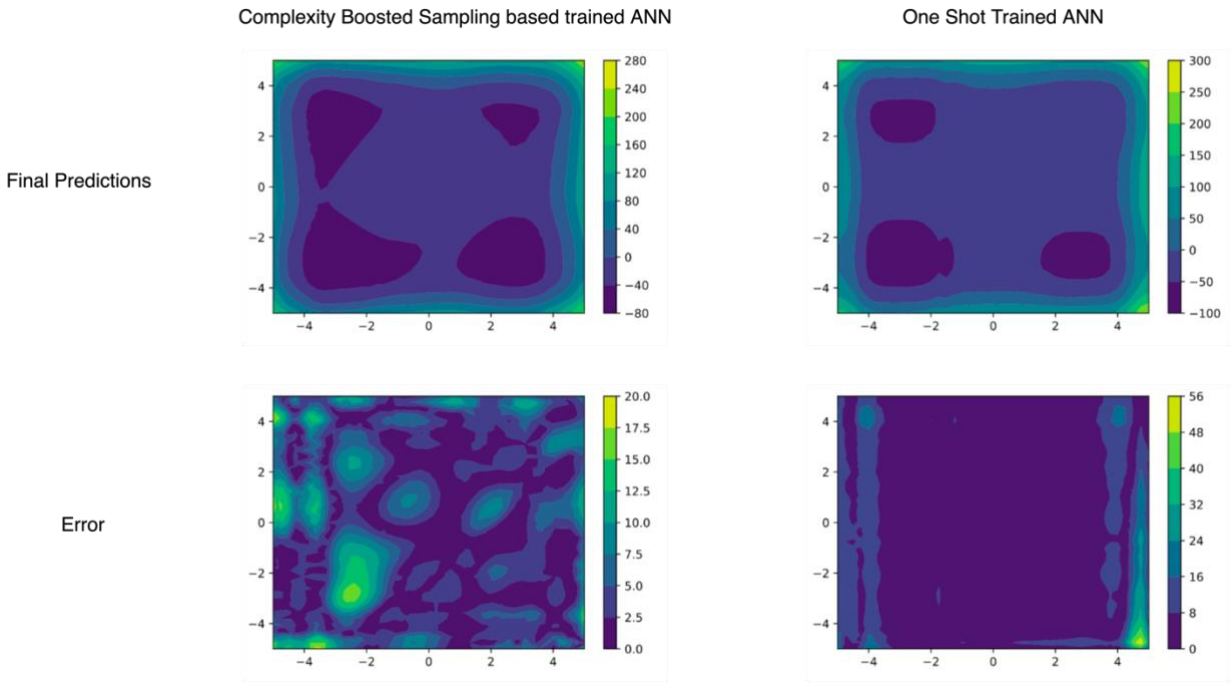


The above set of figures visualizes the training and the sampling process at time steps T = 1,2,3,4.

It can be observed that error keeps decreasing with iterations as new data is generated. The number of points to be sampled in a hypercube depend on the average RMSE value of the hypercube.

We also compared our training methodology with a one-shot training methodology. In one shot training methodology, training data is generated by uniformly sampling points in the whole input space and the neural network is trained for a certain number of iterations on this training data.





As can be observed, complexity boosted sampling based trained ANN achieved a lower error in less iterations on the training set as compared to one shot trained ANN. Both ANN had the same architecture and the same initialized weights.

BUSINESS RELEVANCE

Data sampling task takes good amount of efforts in most of the data driven Machine learning projects, it become challenging when learning is about a system with unknown behavior patterns or the risk of predicting a bad output is critical. For example, learning to understand the behavior of nuclear reactor is of high criticality and high risk than in comparison to learning about say a water heater.

For a complex system large amount of data is required to be sampled for all the input range, thus enabling the neural network to learn and capture all the possible input/output combinations.

Using adaptive sampling the dataset is sampled only densely in the region of maximum inaccuracy, this way of sampling reduces redundant data generation for the region of high accuracy. Saving time and efforts to quickly learn a complex system.

BUSINESS: Jio Platforms	**SITE:**	
DATE OF SUBMISSION:		
DETAILS OF INVENTORS		
Lead Inventor	Name :	Palash Sethi
	Lead / Co-Inventor :	Co-Inventor
	Father's / Husband's Name :	
	Address (Present) :	
	Address (Permanent) :	
	Nationality :	
	PAN No. :	
	Aadhaar No. :	
	Email ID (alternate) :	
	Percentage Contribution % :	50
	Digital Signature Image (JPEG Format):	
Co-Inventor – 2	Name :	Shailesh Kumar
	Lead / Co-Inventor :	Co-Inventor
	Father's / Husband's Name :	
	Address (Present) :	
	Address (Permanent) :	
	Nationality :	
	PAN No. :	
	Aadhaar No. :	
	Email ID (alternate) :	
	Percentage Contribution % :	50
	Digital Signature Image (JPEG Format):	

	Yes √	**No √**	**Details (If Yes, Provide Details)**
Is this invention out of external collaborative research?		√	
Is this invention disclosed to outside party?		√	
Do you plan to present the work in public near future?	√		
Do you have a lab notebook?	√		
Have this invention reduced to practice / prototype?	√		

TITLE OF INVENTION

Non - Linear constrained optimization using neural networks

ABSTRACT OF THE INVENTION (Max 300 Words)

Adoption and implementation of automation technology is rapidly growing in several verticals of industry including Manufacturing, Refineries, Warehousing, Telecom etc. Any automation framework defines (1) Each process-unit and its input-to-output relationship, (2) How the process units interact with each other for the functioning of the end-to-end process. It should also allow us to tune certain control-parameter inputs within each process unit so that a certain business objective for productivity/profitability etc. is met. In this patent we propose a two stage ML based framework to perform constrained optimization on non-linear objective functions to find the optimal control-parameters for a given process-unit. Our approach first uses a hybrid dataset of historic along with synthetic simulation data of a process-unit to train a Neural Network based Digital Twin which accurately captures its input-to-output relationship. Next, using a novel Update-Less Back-Prop algorithm on top of this Digital Twin, we iteratively optimize the control-parameter inputs towards a defined profit based objective function. The objective function incorporates the high-level business inputs for eg. price, cost, and profitability relations.

BACKGROUND OF THE INVENTION

The field in which this patent can be used is Industrial Process Optimization using Neural Networks.

Neural Networks are used as universal function approximators in various industrial use cases. Our patent pertains to the field of process control optimization. Industrial processes are non-linear and complex. The current benchmarks in the industry use linear programming methods for control parameters optimization for minimizing or maximizing an objective function which reflects the business use case such as profits, revenue or cost. Linear Programming based methods are high speed but provide less accuracy as they approximate a non-linear process with linear approximators such as linear regressors. Neural networks can be used to approximate these non-linear processes, and then further can be used to optimize the control parameters using the backpropagation algorithm in a constrained fashion. In this patent, we provide a methodology to use the backpropagation algorithm for performing constrained optimization on control parameters.

The current state of the art methodologies employs neural networks for performing non-linear control parameter optimization using various linear and non-linear optimization methods such as genetic search algorithms. Linear optimization for non-linear objective function gives accurate results only in a certain range while the non-linear methods only use the neural network in forward mode, that is, it only uses neural networks for calculating the outputs from the inputs. And the control parameters are changed further for optimization using various heuristics such as genetic search algorithms. Our work uses the gradient information calculated using backpropagation algorithm to change control parameters to minimize or maximize the objective function formulated for a specific business problem such as profit maximization, while following the constraints of the control parameters such as linear constraints and boundary constraints.

DETAILED DESCRIPTION OF THE INVENTION

1. The patent takes a two-step approach for optimizing control parameters of a process unit
    a. Causality Learning: This step trains a neural network to approximate a process unit accurately. This work is captured in our previous patent “METHOD AND SYSTEM FOR LEARNING BEHAVIOUR OF HIGHLY COMPLEX AND NON-LINEAR SYSTEMS”.
    b. Control Optimisation: This step optimizes process controls in a constrained fashion using backpropagation
2. A process objective function is created which captures the linear and boundary constraints of the process controls.
3. Process controls are iteratively optimised using Adam Optimiser by using backpropagated error. The backpropagation starts at the output layer and gradient of the objective function is calculated with respect to the process controls. Adam optimiser uses the gradient values to change process controls to minimise or maximise the objective function.
4. Process control stops changing when the objective function is optimised

The patent defines a process unit as an industrial system or a mathematical model. The process unit takes a vector or a list of real numbers as an input variable and produces a vector or a list of real numbers as the output. The input is composed of two sub lists, namely:

1. Constants
2. Controls

The constants and controls are concatenated to produce an input vector which can be fed into the process unit to generate output vector. The constants and controls are contextual and depend on the process. During the process of optimization, only the control variables are changed while the constants are kept unchanged. An example of a process unit is a chemical unit in a refinery system. In a chemical unit, the set of control variables can be temperature, pressure, and feed flow, while the constants can be the composition of the input feed. In this system, temperature, pressure and feed flow can be changed iteratively in a constrained manner to optimize and objective function such a profit.

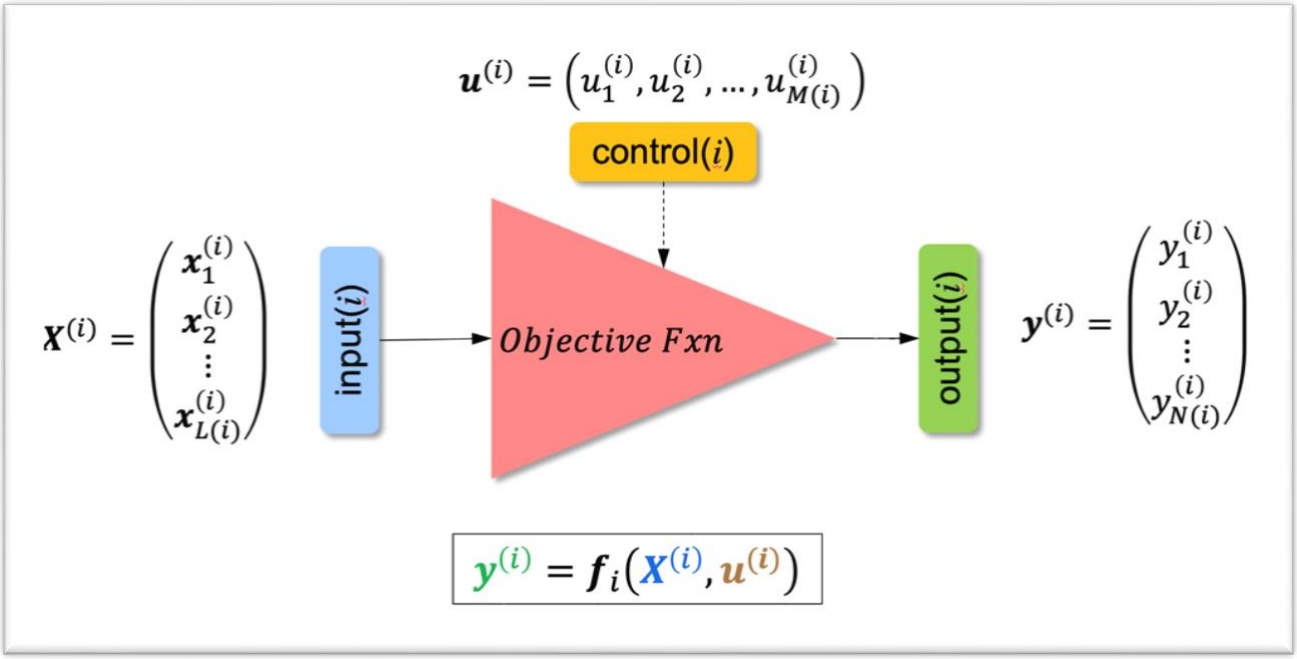


Figure 1 – Process unit

A process unit is defined in Figure 1, where

- **X** is the constant vector
- **u** is the control vector
- **y** is the output vector and is a function of constant vector **X** and control vector **u**
- **i** is the iteration during the process of optimization

X and **u** are concatenated and fed into the process unit to generate **y**

The patent defines the process of optimization of a process unit as a two-step process:

1. Causality Learning
2. Control Parameters Optimization using backpropagation

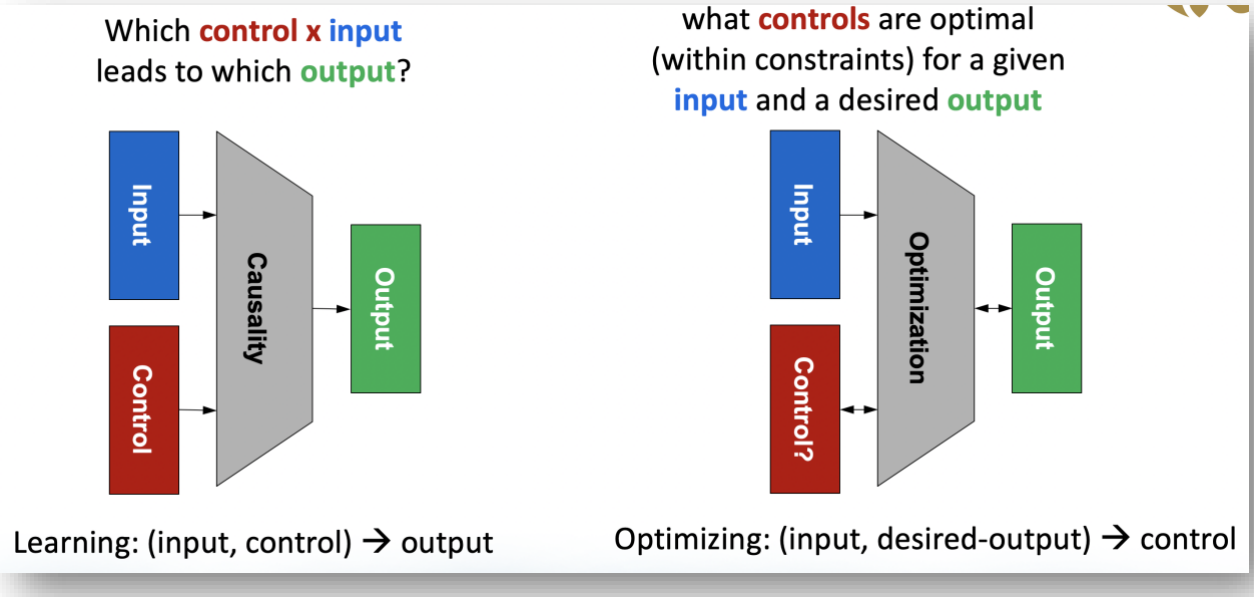


Figure 2 – The two phases of optimization

Figure 2 defines the two phases of optimization

In the phase of causality learning, a neural network is trained on a synthetically generated dataset i.e. a forward mapping is learnt that maps constant parameters and control parameters to the output accurately.

Neural Networks can be trained in two ways:

3. Using pre-collected dataset
4. By sampling mathematical models which can simulate physical systems

We generate synthetic data by adaptive sampling as discussed in our previous patent. If the mathematical model of the physical process is not completely accurate, the synthetic data can be combined with process data from the physical process to generate a hybrid dataset. This hybrid dataset can be further used to fine tune the trained neural network.

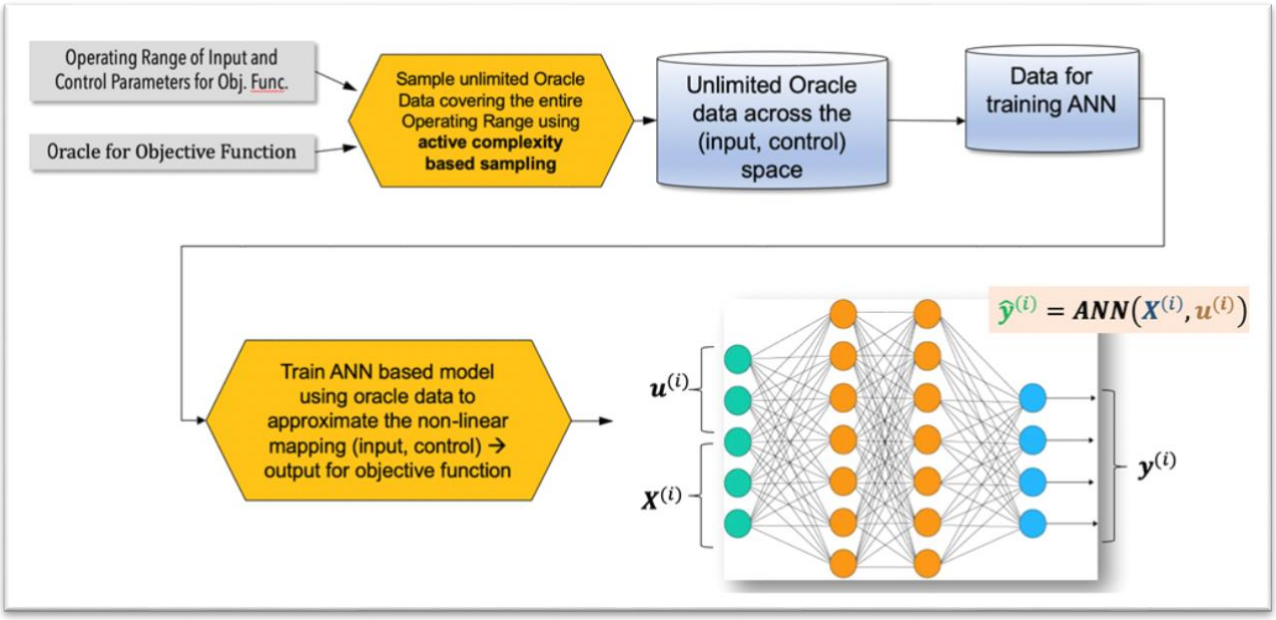


Figure 3 – Training accurate neural network with synthetic data

Figure 3 describes a flowchart of accurately training a neural network with synthetically generated data using active complexity-based sampling.

Once the causality is learned by the Neural Network, it is used to optimize controls for an objective function. Objective function can have multiple constraints for the controls, like, boundary constraints and linear constraints.

Neural Networks can be used to perform optimization by gradient based methods. Modern NN programming uses dynamic computational graphs for maintaining a record of gradients for the parameters of NN. These gradients can be used to perform gradient descent on the controls for optimizing an objective function.

Multiple smaller objective functions can be combined together using Lagrange Multipliers to create a global objective function. These Lagrange Multipliers will act as hyper-parameters for the optimization pipeline.

Following is an example of a generic objective function used to maximize profits while being limited by boundary and linear constraints on the controls side.

Given a Product Revenue **R**, Input Cost **S** and Operation Cost **O** for a Digital Twin, Profit function **J** can be defined as

$$J(f,c) = -(R(f,c) - S(f) - O(c)) - \lambda_{c1}L_{c1}(c) - \lambda_{F1}L_{F1}(f) - \lambda_{c2}L_{c2}(c) - \lambda_{F2}L_{F2}(f)$$

Where

- **f**, **c** are process control parameters
- λ_{c1}, λ_{c2}, λ_{F1} and λ_{F2} are lagrange multipliers
- L_{c1}, L_{c2} , L_{F1} and L_{F2} are constraint functions for **f** and **c** respectively

We consider two constraint functions :

1. Boundary Constraints
2. Linear Constraints

Boundary constraints (L_{c1}, L_{F1}) function provides a min-max range for the process controls to be optimised in, and are defined by

$$L_{c1/F1}(\bar{\boldsymbol{c}}) = \sum_{k} \sigma_\lambda\left(C_k^{min} - c_k, \alpha\right) + \sigma_\lambda(c_k - C_k^{max}, \alpha)$$

- k is the number of variables in the control vector
- C_k^{min} and C_k^{max} are the min-max values for the variable c_k
- $\sigma_\lambda(x, \alpha)$ is a modified sigmoid function and is defined as

$$\sigma_\lambda(x, \alpha) = \frac{1}{1 + e^{-\alpha * x}}$$

- α is a hyperparameter whose value is 500.

Figure 4 gives a conceptual understanding of the boundary constraint function

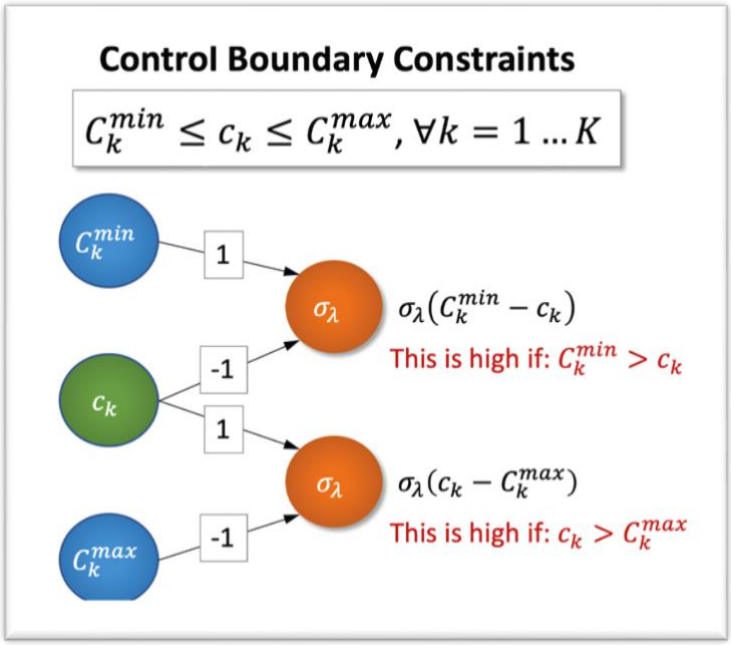


Figure 4 – Boundary Constraint Function

Linear constraint function constraints the process control variables to follow a linear constraint and is defined as

$$L_{c2/F2} = \sum_{k} RMSE(Q - \theta_k c_k)$$

where:

- $RMSE$ is the root mean square error function
- Q is a constant
- θ_k is the coefficient of c_k

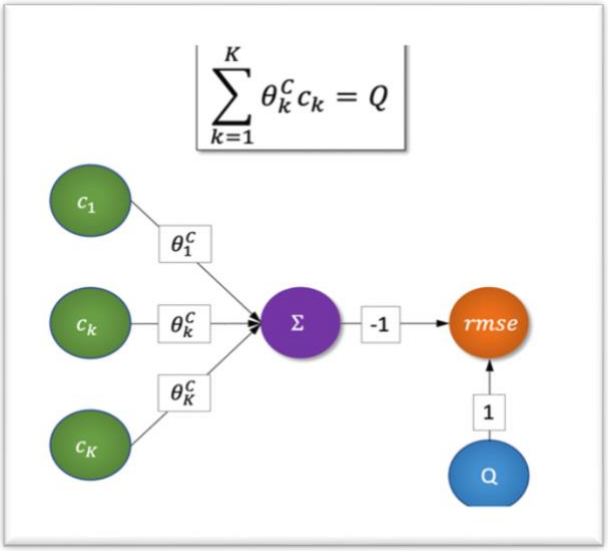


Figure 5 – Linear Constraint Function

Once the objective function is formulated for the business use case and includes the constraint functions, the process of iterative optimization is performed.

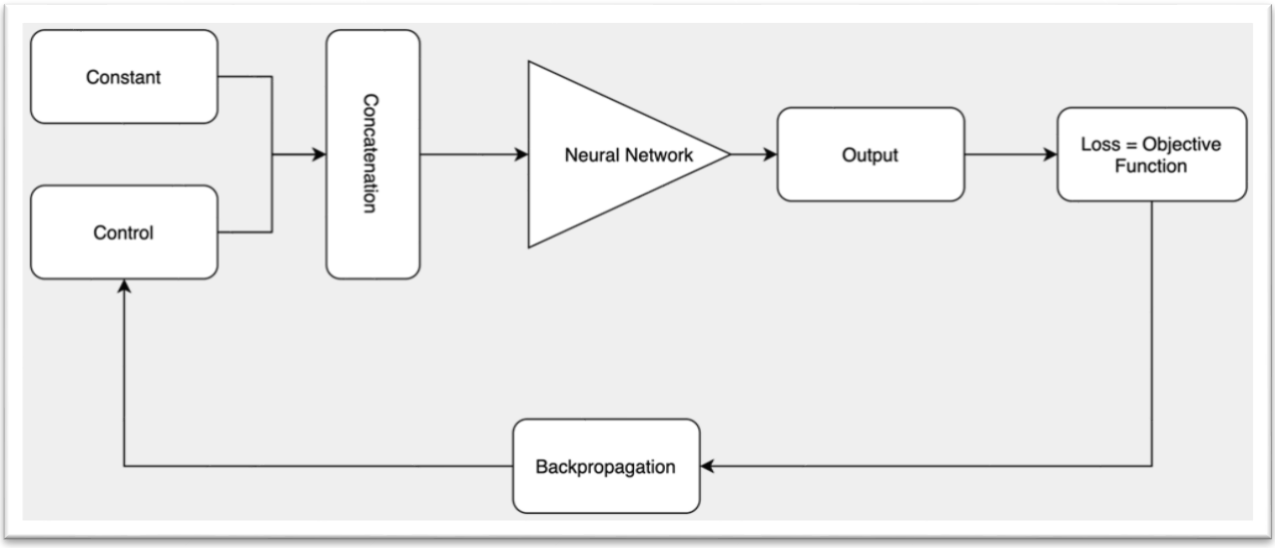


Figure 6 – Process Control Parameter Optimization

Figure 6 gives a brief workflow of the Process Control Parameter Optimization

GIVEN:

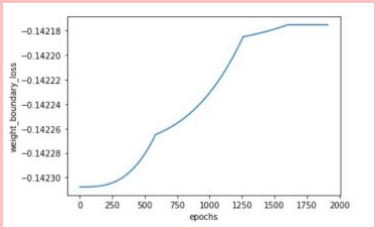
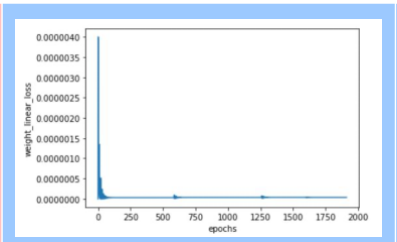
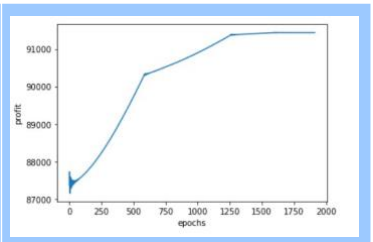
- PRICE of all outputs

- COST of all inputs
- **Trained Neural Network**
- Initial value of input $\bar{f}_0$ and control $\bar{c}_0$ parameters

OPTIMIZATION OF FEED AND CONTROL USING GRADIENT ASCENT

1. Do a forward pass for the current $\bar{f}_t$ and $\bar{c}_t$ and estimate profit = Profit($\bar{f}_t, \bar{c}_t$)
2. The objective function is calculated
3. Loss to be backpropagated is the objective function
4. This error DOES NOT update any of the weights in the already trained neural network
5. This error propagates ALL THE WAY to the control vectors
6. We update the control vectors based on this backpropagated error using Adam Optimizer
7. If maximum percentage change in control/input parameter is < threshold, then stop
8. Otherwise go to step 1

Result :

	Boundary Constraints	Linear Constraints	Profitability
Feed	 $F_n^{min} \le f_n \le F_n^{max}$ $\forall n = 1 \dots N$	 $F_{sum}^{min} \le \sum_{n=1}^{N} \theta_n^F f_n \le F_{sum}^{max}$	 $\mathrm{R}(\bar{f}, \bar{c}) - \mathrm{S}(\bar{f})$

✓ Profitability is maximized ensuring boundary and linear constraints are not violated
✓ Estimated Optimized Profit is highly accurate

The proposed patent helps to create a digital twin of a process unit which can perform constrained optimization of control parameters to minimize or maximize an objective function. Since we use a Neural Network as a digital twin, it can capture non-linearities of the industrial process while the current Industrial Process models try to approximate non-linear process using linear approximation, which are not as accurate as Neural Networks.

The proposed patent creates an end-to-end differentiable digital twin model of a process unit, and uses gradient flows for optimization as compared to other digital twin models that are gradient-free.

The proposed patent and the tool can help in optimising non-linear process units in Reliance, which are currently only being optimised using gradient-free linear methods which do not capture non linearities. An optimization system / model using ANN helps to manage operations in it true nature of state, capturing dynamic interactions of all the measured variables in input and output space. Using such system in plant operations will help reliance realize maximum productivity and efficiency from its assets.

FORM 2

THE PATENTS ACT, 1970

(39 OF 1970)

AND

THE PATENT RULES, 2003

PROVISIONAL SPECIFICATION

(See section 10 and rule 13)

METHOD AND SYSTEM FOR LEARNING BEHAVIOUR OF HIGHLY COMPLEX AND NON-LINEAR SYSTEMS

We, **JIO PLATFORMS LIMITED**, an Indian Citizen of, Office 101, Saffron, Nr. Centre Point, Panchwati 5 Rasta, Ambawadi, Ahmedabad-380006, Gujarat, India.

The following specification describes the invention:

ABSTRACT

METHOD AND SYSTEM FOR LEARNING BEHAVIOUR OF HIGHLY COMPLEX AND NON-LINEAR SYSTEMS

The present disclosure generally relate to handling data of non-linear, multi-variable complex system. More particularly, the present disclosure relates to methods and systems for training machine learning-based computing devices to ensure adaptive sampling of highly complex data packets. The present invention provides a robust and effective solution to implement a complexity-based sampling which trains the Neural Network in complex mapping regions, by iteratively sampling the oracle and training the neural network in complex regions. A Machine Learning (ML) or Artificial intelligence (AI) models may be built to solve the problem efficiently. The present invention involves a complexity-based sampling methodology to train a neural Network via a third set of instructions

METHOD AND SYSTEM FOR LEARNING BEHAVIOUR OF HIGHLY COMPLEX AND NON-LINEAR SYSTEMS

FIELD OF INVENTION

[0001] The embodiments of the present disclosure generally relate to handling data of non-linear, multi-variable complex system. More particularly, the present disclosure relates to methods and systems for training machine learning-based computing devices to ensure adaptive sampling of highly complex data packets.

BACKGROUND OF THE INVENTION

[0002] The following description of related art is intended to provide background information pertaining to the field of the disclosure. This section may include certain aspects of the art that may be related to various features of the present disclosure. However, it should be appreciated that this section be used only to enhance the understanding of the reader with respect to the present disclosure, and not as admissions of prior art.

[0003] In general, Neural Networks need large amount of training data for learning non-linear, multi-variable complex system. In principle the Neural Networks can be trained to approximate system's behavior with large set of training data over the entire input range. But generating data for the entire input range is cumbersome and time consuming.

[0004] Conventional methods for Non-Linear Complex simulations of physical systems use the laws of physics such as thermodynamics, chemical reactions, differential equations etc. are simulated to solve for a given input. These simulations are time-taking and are unfeasible to run multiple times for the entire operational range in business situations. Linear Approximation of the complex simulations help in faster results generation with acceptable errors, it is less accurate but quick to do and works only for a small range of input domain.

[0005] Neural Networks can be trained in two ways: Using pre-collected dataset and by sampling mathematical models which can simulate physical systems. For training Neural Networks to approximate a black-box model, there are no existing methods wherein the simulation dataset is sampled using a black-box model, or via an Oracle function that can efficiently simulate the given physical system. Moreover, the sampling is done linearly and not optimized in such a way that more data is sampled where the complexity of the manifold is high, while keeping the number of total data points sampled at minimum. Even if an oracle acquires this training data by sampling the points in a given input domain and running this set of input data through the oracle to generate outputs, the sampling methodology applied for this process is uniform random sampling. Uniform random sampling assumes that all points in the input domain are equally significant for the learning task, which, is not sufficient to map highly complex input-output mapping.

[0006] There is therefore a need in the art to provide a method and system that can overcome the shortcomings of the existing prior art.

OBJECTS OF THE PRESENT DISCLOSURE

[0007] Some of the objects of the present disclosure, which at least one embodiment herein satisfies are as listed herein below.

[0008] An object of the present disclosure is to provide for a method and a system that facilitates adopting smart sampling of large input space with focused approach on areas of input space with low accuracy.

[0009] An object of the present disclosure is to provide for a method and a system that facilitates capturing of maximum non-linearity of a system and providing good representation in neural networks.

[0010] An object of the present disclosure is to provide for a method and a system that requires minimum amount of training data.

[0011] An object of the present disclosure is to provide for a method and a system that facilitates a higher accurate training model.

[0012] An object of the present disclosure is to provide for a method and a system that mimics the human learning process.

BRIEF DESCRIPTION OF DRAWINGS

[0013] The accompanying drawings, which are incorporated herein, and constitute a part of this invention, illustrate exemplary embodiments of the disclosed methods and systems in which like reference numerals refer to the same parts throughout the different drawings. Components in the drawings are not necessarily to scale, emphasis instead being placed upon clearly illustrating the principles of the present invention. Some drawings may indicate the components using block diagrams and may not represent the internal circuitry of each component. It will be appreciated by those skilled in the art that invention of such drawings includes the invention of electrical components, electronic components or circuitry commonly used to implement such components.

[0014] FIG. 1 illustrates an exemplary network architecture in which or with which proposed system of the present disclosure can be implemented, in accordance with an embodiment of the present disclosure.

[0015] FIG. 2 illustrates an exemplary representation of proposed system / machine learning (ML) engine for training a neural network, in accordance with an embodiment of the present disclosure.

[0016] FIG. 3 illustrates an exemplary flow diagram representation of a proposed method, in accordance with an embodiment of the present disclosure.

[0017] FIG. 4 illustrates an exemplary block representation of detailed sampling method, in accordance with an embodiment of the present disclosure.

[0018] FIGs. 5A-5C illustrate exemplary block diagram representations of hypercubes, in accordance with an embodiment of the present disclosure.

[0019] FIGs. 6A-6E illustrates exemplary representations of the analysis of the proposed method, in accordance with an embodiment of the present disclosure.

[0020] FIG. 7 illustrates an exemplary computer system in which or with which embodiments of the present invention can be utilized, in accordance with embodiments of the present disclosure.

[0021] The foregoing shall be more apparent from the following more detailed description of the invention.

DETAILED DESCRIPTION OF INVENTION

[0022] In the following description, for the purposes of explanation, various specific details are set forth in order to provide a thorough understanding of embodiments of the present disclosure. It will be apparent, however, that embodiments of the present disclosure may be practiced without these specific details. Several features described hereafter can each be used independently of one another or with any combination of other features. An individual feature may not address all of the problems discussed above or might address only some of the problems discussed above. Some of the problems discussed above might not be fully addressed by any of the features described herein.

[0023] The ensuing description provides exemplary embodiments only, and is not intended to limit the scope, applicability, or configuration of the disclosure. Rather, the ensuing description of the exemplary embodiments will provide those skilled in the art with an enabling description for implementing an exemplary embodiment. It should be understood that various changes may be made in the function and arrangement of elements without departing from the spirit and scope of the invention as set forth.

[0024] The present invention provides a robust and effective solution to implement a complexity-based sampling which trains the Neural Network in complex mapping regions, by iteratively sampling the oracle and training the neural network in complex regions. A Machine Learning (ML) or Artificial intelligence (AI) models may be built to solve the problem efficiently. The present invention involves a complexity-based sampling methodology to train a neural Network via a third set of instructions.

[0025] Referring to FIG. 1 that illustrates an exemplary network architecture for training a neural network-based computing (100) (also referred to as system (100)) in which or with which a system (110) or simply referred to as the system (110) of the present disclosure can be implemented, in accordance with an embodiment of the present disclosure. As illustrated, the exemplary architecture (100) may be equipped with a machine learning (ML)) engine (214) for facilitating training of the system. The system may receive a set of data packets from a plurality of first computing devices (104-1, 104-2…104-N) associated with users (102-1, 102-2, 102-3…102-N) (individually referred to as the user (102) or the employer (102) and collectively referred to as the users (102) or the employers (102)). The system (110) may be further operatively coupled to a second computing device (108) associated with an entity (114). The entity (114) may include a company, a university, a lab facility, a business enterprise, a defence facility, or any other secured facility. The system (110) may be communicatively coupled to the one or more first computing devices (individually referred to as the first computing device (104) and collectively referred to as the first computing devices (104)).

[0026] In an exemplary embodiment, the first computing devices (104) may include non linear and complex physical systems performing complex physical or chemical process but not limited to the like. Examples of non-linear and complex physical systems may be nuclear reactors and the like. Learning to understand the behavior of but not limited to nuclear reactor is of high criticality and high risk than in comparison in learning about a water heater and the like.

[0027] The system (110) may be coupled to a centralized server (112). The centralized server (112) may also be operatively coupled to the one or more first computing devices (104) and the second computing devices (108) through a communication network (106).

[0028] In an embodiment, system (110) may execute, a first set of instructions (interchangeably referred to as curriculum sampling) through the ML engine (214) on the received set of data packets pertaining to determining a complexity of a region in the set of data packets received and then sample by the first set of instructions a plurality of samples points proportional to the complexity of the region in a plurality of iterations. The system (110) may further be configured to determine by a second set of instructions (interchangeably referred to as regression tree) in the ML engine (214) a plurality of regions of constant complexity. In a way of example and not as a limitation, the regression tree may include KD Trees and the like.

[0029] The system (110) may then train the neural network by a third set of instructions (also referred to as the oracle function), wherein the third set of instructions pertain to generating weights for each iteration after which the system (110) may execute the third set of instructions on the neural network until a predefined threshold is reached. The predefined threshold may pertain to accuracy of the trained neural network in the region for the iteration.

[0030] In a way of example and not as a limitation, the oracle function may attempt to simulate a complex physical or chemical process with an objective function. The oracle function is often supported with domain specific tools, such as a simulation software. A multidimensional input range is selected based on domain knowledge.

[0031] In an embodiment, the network architecture (100) may be modular and flexible to accommodate any kind of changes in the system (110) as proximate processing may be acquired towards training the neural network. The system (110) configuration details can be modified on the fly.

[0032] In an exemplary embodiment, the architecture of the neural network may be initialized with random weights and biases and then fed in as a hyperparameter to a training pipeline associated with the system (110).

[0033] In an exemplary embodiment, a communication network (106) may include, by way of example but not limitation, at least a portion of one or more networks having one or more nodes that transmit, receive, forward, generate, buffer, store, route, switch, process, or a combination thereof, etc. one or more messages, packets, signals, waves, voltage or current levels, some combination thereof, or so forth. A network may include, by way of example but not limitation, one or more of: a wireless network, a wired network, an internet, an intranet, a public network, a private network, a packet-switched network, a circuit-switched network, an ad hoc network, an infrastructure network, a Public-Switched Telephone Network (PSTN), a cable network, a cellular network, a satellite network, a fiber optic network, some combination thereof.

[0034] In another exemplary embodiment, the centralized server (112) may include or comprise, by way of example but not limitation, one or more of: a stand-alone server, a server blade, a server rack, a bank of servers, a server farm, hardware supporting a part of a cloud service or system, a home server, hardware running a virtualized server, one or more processors executing code to function as a server, one or more machines performing server-side functionality as described herein, at least a portion of any of the above, some combination thereof.

[0035] In an embodiment, the one or more first computing devices (104), the one or more second computing devices (108) may communicate with the system (110) via set of executable instructions residing on any operating system, including but not limited to, Android TM, iOS TM, Kai OS TM and the like. In an embodiment, to one or more first computing devices (104), and the one or more second computing devices (108) may include, but not limited to, any electrical, electronic, electro-mechanical or an equipment or a combination of one or more of the above devices such as mobile phone, smartphone, Virtual Reality (VR) devices, Augmented Reality (AR) devices, laptop, a general-purpose computer, desktop, personal digital assistant, tablet computer, mainframe computer, or any other computing device, wherein the computing device may include one or more in-built or externally coupled accessories including, but not limited to, a visual aid device such as camera, audio aid, a microphone, a keyboard, input devices for receiving input from a user such as touch pad, touch enabled screen, electronic pen, receiving devices for receiving any audio or visual signal in any range of frequencies and transmitting devices that can transmit any audio or visual signal in any range of frequencies. It may be appreciated that the to one or more first computing devices (104), and the one or more second computing devices (108) may not be restricted to the mentioned devices and various other devices may be used. A smart computing device may be one of the appropriate systems for storing data and other private/sensitive information.

[0036] FIG. 2 with reference to FIG. 1, illustrates an exemplary representation of system (110)/Machine Learning (ML) engine (214) for facilitating training of a neural network, in accordance with an embodiment of the present disclosure. In an aspect, the system (110) /ML engine (214) may comprise one or more processor(s) (202). The one or more processor(s) (202) may be implemented as one or more microprocessors, microcomputers, microcontrollers, edge or fog microcontrollers, digital signal processors, central processing units, logic circuitries, and/or any devices that process data based on operational instructions. Among other capabilities, the one or more processor(s) (202) may be configured to fetch and execute computer-readable instructions stored in a memory (204) of the system (110). The memory (204) may be configured to store one or more computer-readable instructions or routines in a non-transitory computer readable storage medium, which may be fetched and executed to create or share data packets over a network service. The memory (204) may comprise any non-transitory storage device including, for example, volatile memory such as RAM, or non-volatile memory such as EPROM, flash memory, and the like.

[0037] In an embodiment, the system (110)/ML engine (214) may include an interface(s) 206. The interface(s) (206) may comprise a variety of interfaces, for example, interfaces for data input and output devices, referred to as I/O devices, storage devices, and the like. The interface(s) (206) may facilitate communication of the system (110). The interface(s) (206) may also provide a communication pathway for one or more components of the system (110) or the centralized server (112). Examples of such components include, but are not limited to, processing unit/engine(s) (208) and a database (210).

[0038] The processing unit/engine(s) (208) may be implemented as a combination of hardware and programming (for example, programmable instructions) to implement one or more functionalities of the processing engine(s) (208). In examples described herein, such combinations of hardware and programming may be implemented in several different ways. For example, the programming for the processing engine(s) (208) may be processor

executable instructions stored on a non-transitory machine-readable storage medium and the hardware for the processing engine(s) (208) may comprise a processing resource (for example, one or more processors), to execute such instructions. In the present examples, the machine-readable storage medium may store instructions that, when executed by the processing resource, implement the processing engine(s) (208). In such examples, the system (110)/centralized server (112) may comprise the machine-readable storage medium storing the instructions and the processing resource to execute the instructions, or the machine-readable storage medium may be separate but accessible to the system (110)/centralized server (112) and the processing resource. In other examples, the processing engine(s) (208) may be implemented by electronic circuitry.

[0039] The processing engine (208) may include one or more engines selected from any of a data acquisition engine (212), ML engine (214), and other engines (216). The processing engine (208) may further be for complex sampling processing but not limited to the like.

[0040] FIG. 3 illustrates an exemplary flow diagram representation of a proposed method, in accordance with an embodiment of the present disclosure. As illustrated, the method (300) for training a neural network may include at 302, the step of receiving a set of data packets pertaining to input generated by non-linear, multi-variable complex computing devices and at 304, the step of executing, at the processor, a first set of instructions on the received set of data packets, the first set of instructions pertaining to determining a complexity of a region in the set of data packets received.

[0041] The method (300) may further include at 306, the step of sampling, at the processor by the first set of instructions a plurality of samples points proportional to the complexity of the region in a plurality of iterations and at 308, the step of determining, at the processor by a second set of instructions a plurality of regions of constant complexity.

[0042] Furthermore, the method (300) may include at 310, the step of training the neural network, at the processor by a third set of instructions. The third set of instructions pertain to generating weights for each iteration and at step 312, the step of executing the third set of instructions on the network until a predefined threshold is reached, wherein the predefined threshold pertains to accuracy of the trained neural network in the region for the iteration.

[0043] FIG. 4 illustrates an exemplary block representation of detailed sampling method, in accordance with an embodiment of the present disclosure. As illustrated, in an aspect, at block 402 uniform random sampling is performed in the given input range, where the number of samples generated per iteration is a hyperparameter to the training pipeline. The input samples may be fed into the oracle function in a parallelizable fashion and outputs for the sampled inputs may be generated. The outputs generated may be divided into training dataset (406) and test dataset (404). λ, the batch importance parameter assigned to this initial training set is at least but not limited to 1.0.

[0044] In an exemplary embodiment, the training process begins by choosing an optimizer and its learning rate. The optimizer used can be but not limited to Adam Optimizer. The neural network may be trained on the sampled dataset for a given number of epochs. The loss function used to train the neural network may be defined as:

$$\frac{1}{\max(i)} \sum_{i} \frac{1}{N} \sum_{j} \lambda^{(i-\max(i))} * \left(y_{ij} - \bar{y}_{ij} \right)^2$$

where i is the sampling iteration

j is the sample in i^{th} iteration

$\max(i)$ is the current sampling iteration

N is the number of samples generated per iteration

λ is the batch importance parameter

y_{ij} and $\bar{y}_{ij}$ are the target and predicted output values

[0045] In an exemplary embodiment, the trained neural network (408) may be tested on the training data (406), and mean squared error may be calculated for each sample in the training data. A tested neural network (410) may be first checked for accuracy. At block 412, if accuracy is greater than a first predefined threshold or if change in value is less than a second predefined threshold value, train a decision tree at block 414. Since no geometry could capture actual error cloud, n-dimensional hypercubes (416) may be applied to approximate an error distribution in the space and the decision trees may be used to identify and sample the hypercubes (418).

[0046] In an exemplary embodiment, a regression tree may be trained on the sampled input data against the mean squared error of outputs predicted by the neural network. This is done to identify pure regions in the error domain. These regions have almost constant error values. To identify these regions, a depth first search algorithm may be implemented on the trained decision tree, and the decision rules leading to leaf nodes may be identified. These decision rules identify pure n-dimensional hypercubes.

[0047] FIGs. 5A-5C illustrate exemplary block diagram representations of hypercubes, in accordance with an embodiment of the present disclosure. As illustrated, a hypercube may include a dataset at block 502 is sent to fit decision tree block 506 governed by a predefined decision tree algorithm at block 504. The fit decision tree may e also fed with errors at block 508, after the errors have undergone Z score normalization at block 510.

[0048] In an exemplary embodiment, the hypercube may be defined by the volume, the number of points and the average error value associated to the points in the encompassed region and may be given by

$$hypercube : \{n_k^t, e_k^t, v_k^t\}_k^L$$

where n_k^t, e_k^t, v_k^t are the number of points, their average error and the volume of k^{th} hypercube, L is the number of leaf nodes learnt by the regression tree.

[0049] In an exemplary embodiment the normalizing parameter, Z is calculated for the entire set of hypercubes

$$Z = \sum_k (e_k^t \times v_k^t)^\alpha$$

where α is a hyperparameter required for exponential sampling numbers

[0050] As illustrated in FIG. 5B, a trained decision tree at block 522 and each data traverses the decision tree at block 524 to generate leaf nodes. At each leaf node at block 526, space range occupied by the leaf node is calculated at block 530 and volume of space is calculated at block 532. At block 534, points residing in the leaf node space are filtered by receiving the data set from block 502 and the errors at block 508 after which either root mean square error in each node is calculated at block 536 or at block 540, calculate the number of points present in the leaf node. From blocks 532, 536 and 540, hypercubes are generated at block 538. Then, if all leaf nodes have been traversed are checked. If not, the procedure starts again. If traversed, the procedure is stopped.

[0051] As illustrated in FIG 5C, at block 552, Z=0 is initialized for each hypercube at block 554. At block 556 Error is stored in *e* and volume in *v* in block 558 which are given to block 560 where $Z = Z + \sum_k (e_k^t \times v_k^t)^\alpha$ and updated for each hypercube at block 564 along with the value of α at block 560. A target density for each hypercube may be calculated $p_k^t = \frac{(e_k^t \times v_k^t)^\alpha}{Z}$ at block 566. New samples may be uniformly sampled in each hypercube, with the number of samples being proportional to the target density of the hypercube, that is, spread and intensity of the error values in a pure region decides the total number of points to be sampled at block 570 in the iteration and given by $N_k^t = N \times p_k^t$ at block 568.

[0052] In an exemplary embodiment, the new samples may be passed into the oracle function to generate outputs. The input and output points generated across the hypercubes may be concatenated to create the second batch of training data. The training data may be given a batch importance parameter proportional to the iteration number. The training process may be continued until the neural network achieves an acceptable error on the test dataset.

[0053] FIGs. 6A-6E illustrates exemplary representations of the analysis of the proposed method, in accordance with an embodiment of the present disclosure. As illustrated in FIGs. 6A-6E in a way of example and not as a limitation, a use case scenario using but not limited to "Styblinski Tang" objective function with visualization of accuracy loss in a sample space are shown below. FIG. 6A shows an oracle Visualization for the complete input space. The 'Styblinski Tang' is a benchmark objective function used for testing optimization algorithms. It consists of 4 local minima and 1 global minima. Although the objective function can be used for multi-dimension input, as an example, a 2-D version of the objective function for visualizing various iterations of the training algorithm. As can be seen FIGs 6A-6E, the objective function is highly non-linear near the manifold of local minimas.

[0054] FIGs. 6B and 6C visualize the training and the sampling process at time steps T = 1,2,3,4. It can be observed that the error keeps decreasing with iterations as new data is generated. The number of points to be sampled in a hypercube depend on the average RMSE value of the hypercube. FIG. 6D illustrated comparison of training methodology with a one-shot training methodology. In one shot training methodology, training data is generated by uniformly sampling points in the whole input space and the neural network is trained for a certain number of iterations on this training data. As can be observed from FIG. 6E, complexity boosted sampling based trained ANN achieved a lower error in less iterations on the training set as compared to one shot trained ANN. Both ANN had the same architecture and the same initialized weights.

[0055] FIG. 7 illustrates an exemplary computer system in which or with which embodiments of the present invention can be utilized in accordance with embodiments of the present disclosure. As shown in FIG. 7, computer system 700 can include an external storage device 710, a bus 720, a main memory 730, a read only memory 740, a mass storage device 750, communication port 760, and a processor 770. A person skilled in the art will appreciate that the computer system may include more than one processor and communication ports. Examples of processor 770 include, but are not limited to, an Intel® Itanium® or Itanium 2 processor(s), or AMD® Opteron® or Athlon MP® processor(s), Motorola® lines of processors, FortiSOC™ system on chip processors or other future processors. Processor 770 may include various modules associated with embodiments of the present invention. Communication

port 760 can be any of an RS-232 port for use with a modem based dialup connection, a 10/100 Ethernet port, a Gigabit or 10 Gigabit port using copper or fiber, a serial port, a parallel port, or other existing or future ports. Communication port 760 may be chosen depending on a network, such a Local Area Network (LAN), Wide Area Network (WAN), or any network to which computer system connects. Memory 730 can be Random Access Memory (RAM), or any other dynamic storage device commonly known in the art. Read-only memory 740 can be any static storage device(s) e.g., but not limited to, a Programmable Read Only Memory (PROM) chips for storing static information e.g., start-up or BIOS instructions for processor 770. Mass storage 750 may be any current or future mass storage solution, which can be used to store information and/or instructions. Exemplary mass storage solutions include, but are not limited to, Parallel Advanced Technology Attachment (PATA) or Serial Advanced Technology Attachment (SATA) hard disk drives or solid-state drives (internal or external, e.g., having Universal Serial Bus (USB) and/or Firewire interfaces), e.g. those available from Seagate (e.g., the Seagate Barracuda 782 family) or Hitachi (e.g., the Hitachi Deskstar 7K800), one or more optical discs, Redundant Array of Independent Disks (RAID) storage, e.g. an array of disks (e.g., SATA arrays), available from various vendors including Dot Hill Systems Corp., LaCie, Nexsan Technologies, Inc. and Enhance Technology, Inc.

[0056] Bus 720 communicatively couples processor(s) 770 with the other memory, storage and communication blocks. Bus 720 can be, e.g. a Peripheral Component Interconnect (PCI) / PCI Extended (PCI-X) bus, Small Computer System Interface (SCSI), USB or the like, for connecting expansion cards, drives and other subsystems as well as other buses, such a front side bus (FSB), which connects processor 770 to software system.

[0057] Optionally, operator and administrative interfaces, e.g. a display, keyboard, and a cursor control device, may also be coupled to bus 720 to support direct operator interaction with a computer system. Other operator and administrative interfaces can be provided through network connections connected through communication port 760. The external storage device 710 can be any kind of external hard-drives, floppy drives, IOMEGA® Zip Drives, Compact Disc - Read Only Memory (CD-ROM), Compact Disc-Re-Writable (CD-RW), Digital Video Disk-Read Only Memory (DVD-ROM). Components described above are meant only to exemplify various possibilities. In no way should the aforementioned exemplary computer system limit the scope of the present disclosure.

[0058] Thus, the present disclosure provides for a unique and efficient method for data sampling in a system with unknown behavior patterns or the risk of predicting a bad output. For example, learning to understand the behavior of nuclear reactor is of high criticality and high risk than in comparison to learning about say a water heater. For a complex system large amount of data is required to be sampled for all the input range, thus enabling the neural network to learn and capture all the possible input/output combinations. Using adaptive sampling the dataset is sampled only densely in the region of maximum inaccuracy, this way of sampling reduces redundant data generation for the region of high accuracy, saving time and efforts to quickly learn a complex system.

[0001] While considerable emphasis has been placed herein on the preferred embodiments, it will be appreciated that many embodiments can be made and that many changes can be made in the preferred embodiments without departing from the principles of the invention. These and other changes in the preferred embodiments of the invention will be apparent to those skilled in the art from the disclosure herein, whereby it is to be distinctly

understood that the foregoing descriptive matter to be implemented merely as illustrative of the invention and not as limitation.

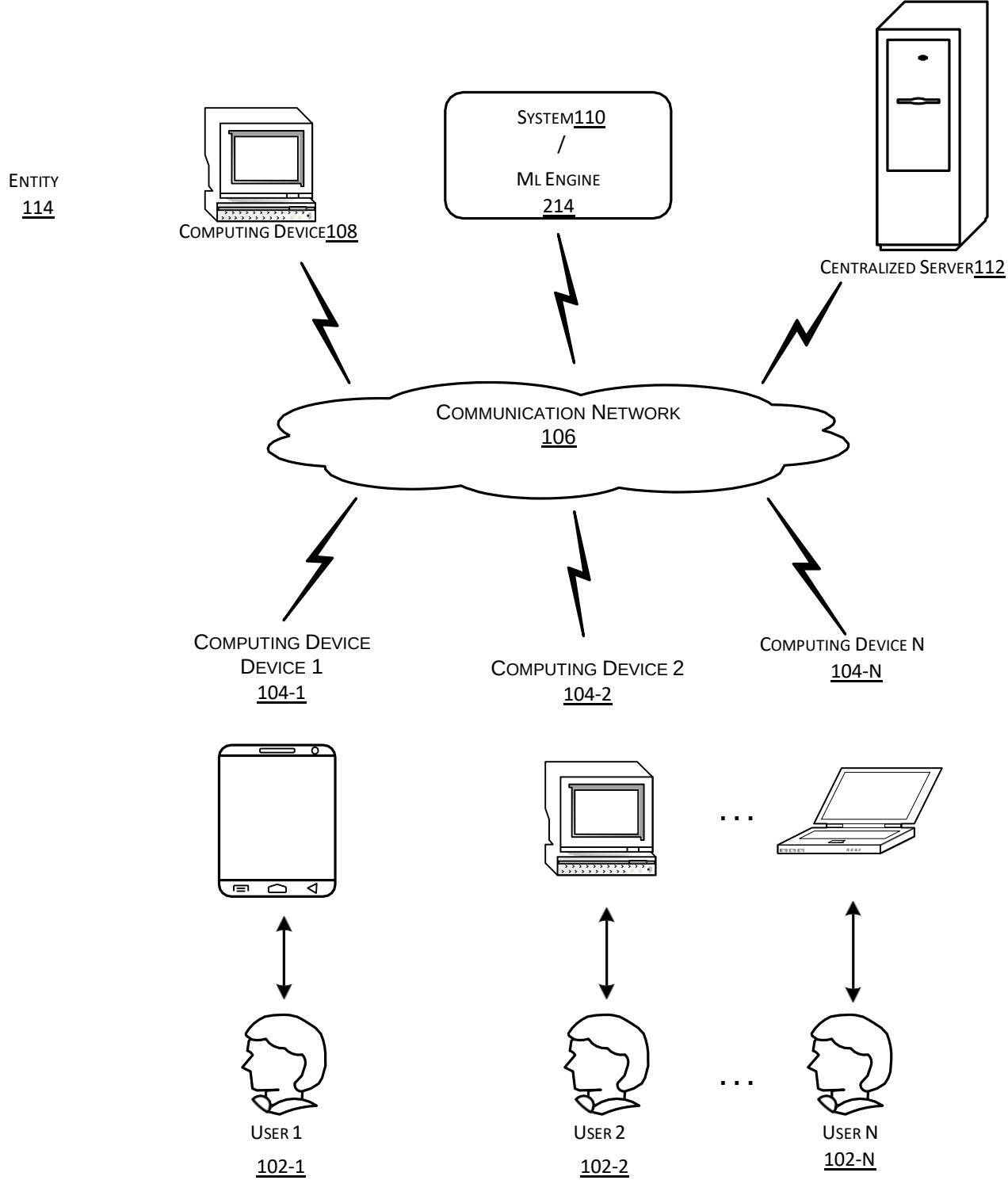

ENTITY
114
SYSTEM110
/
ML ENGINE
214
COMPUTING DEVICE108
CENTRALIZED SERVER112
COMMUNICATION NETWORK
106
COMPUTING DEVICE
DEVICE 1
104-1
COMPUTING DEVICE 2
104-2
COMPUTING DEVICE N
104-N
USER 1
102-1
USER 2
102-2
USER N
102-N

FIG. 1

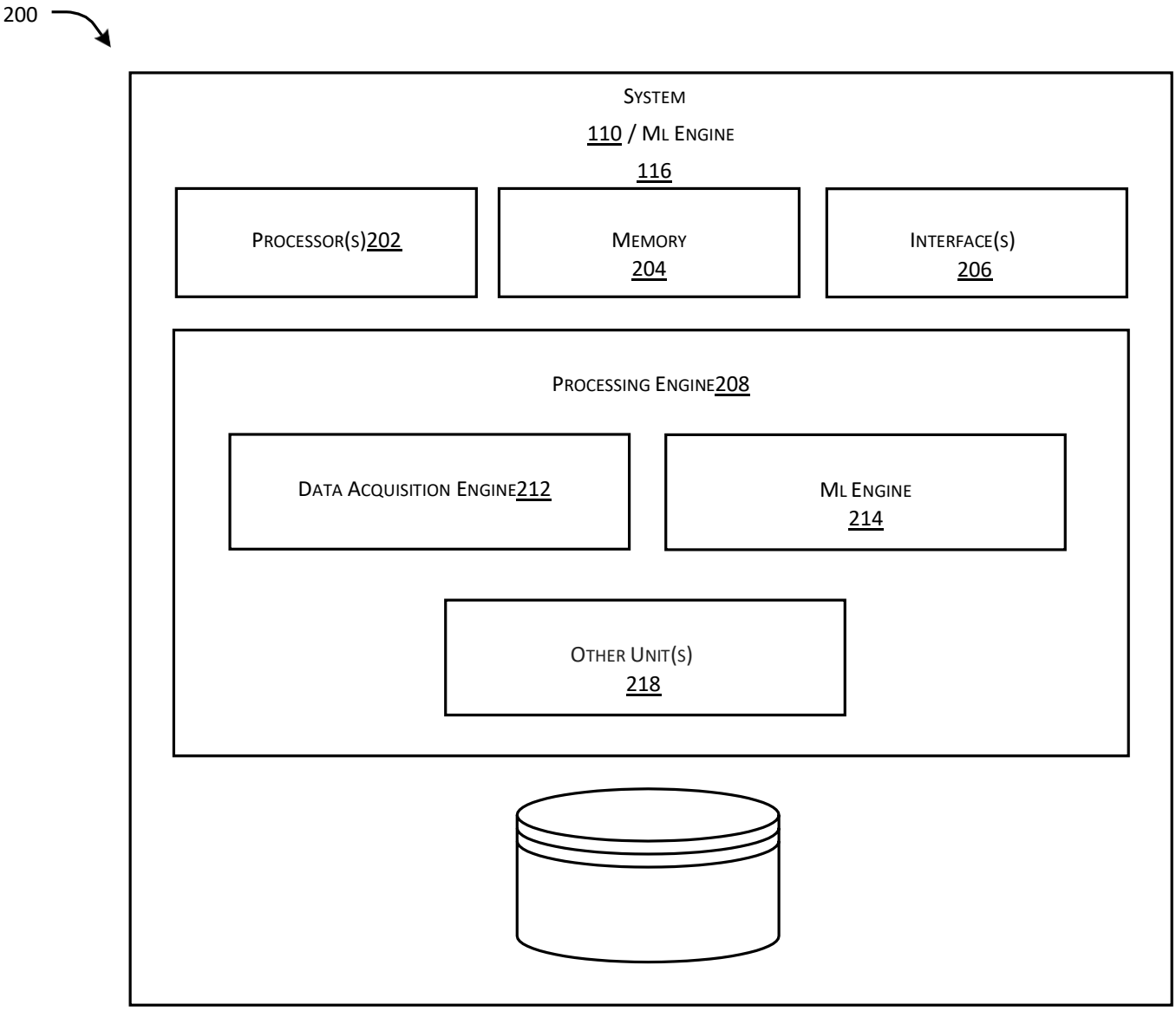

200
SYSTEM
110 / ML ENGINE
116
PROCESSOR(S)202
MEMORY
204
INTERFACE(S)
206
PROCESSING ENGINE208
DATA ACQUISITION ENGINE212
ML ENGINE
214
OTHER UNIT(S)
218

FIG. 2

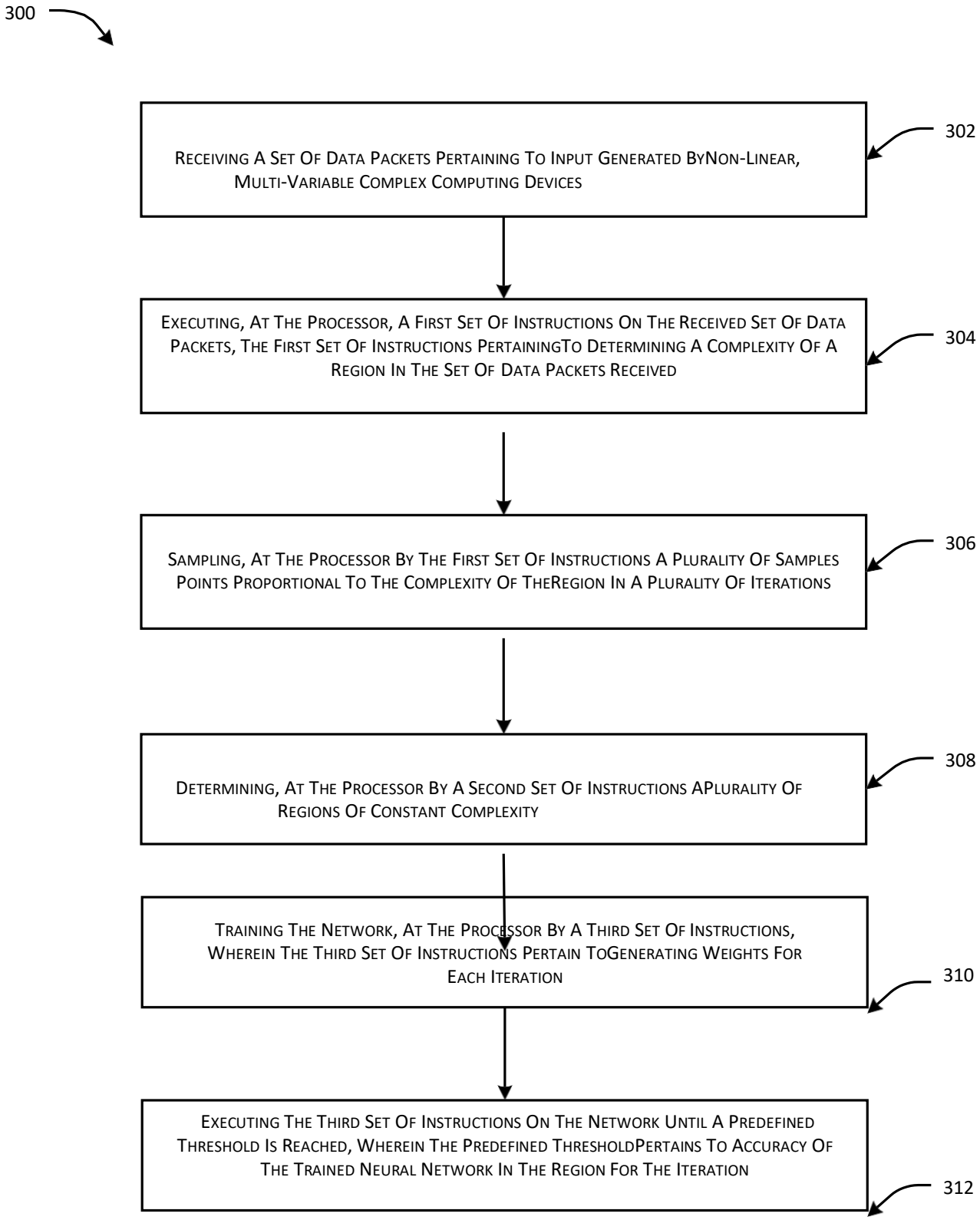
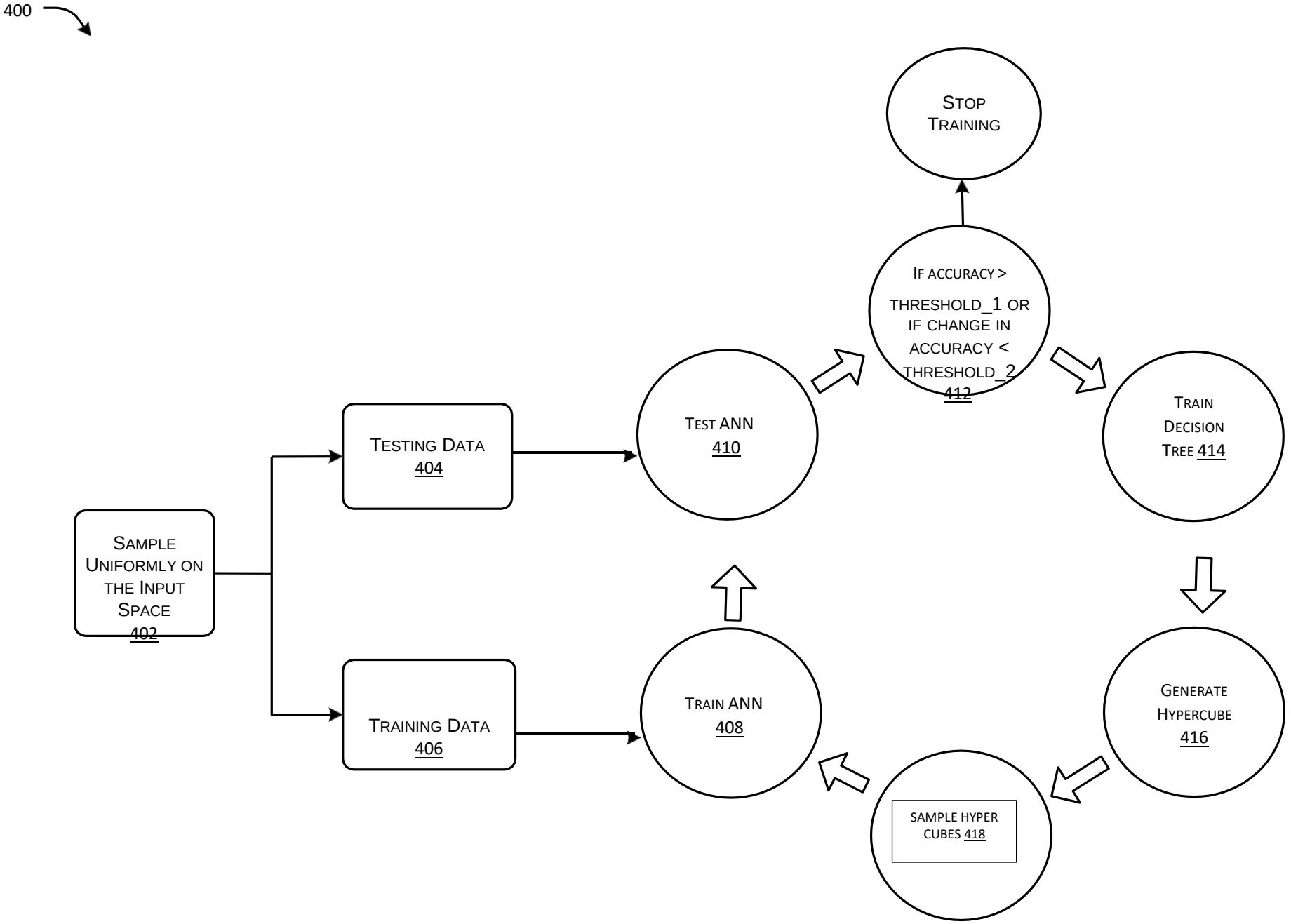

300
RECEIVING A SET OF DATA PACKETS PERTAINING TO INPUT GENERATED BYNON-LINEAR, MULTI-VARIABLE COMPLEX COMPUTING DEVICES
302
EXECUTING, AT THE PROCESSOR, A FIRST SET OF INSTRUCTIONS ON THE RECEIVED SET OF DATA PACKETS, THE FIRST SET OF INSTRUCTIONS PERTAININGTO DETERMINING A COMPLEXITY OF A REGION IN THE SET OF DATA PACKETS RECEIVED
304
SAMPLING, AT THE PROCESSOR BY THE FIRST SET OF INSTRUCTIONS A PLURALITY OF SAMPLES POINTS PROPORTIONAL TO THE COMPLEXITY OF THEREGION IN A PLURALITY OF ITERATIONS
306
DETERMINING, AT THE PROCESSOR BY A SECOND SET OF INSTRUCTIONS APLURALITY OF REGIONS OF CONSTANT COMPLEXITY
308
TRAINING THE NETWORK, AT THE PROCESSOR BY A THIRD SET OF INSTRUCTIONS, WHEREIN THE THIRD SET OF INSTRUCTIONS PERTAIN TOGENERATING WEIGHTS FOR EACH ITERATION
310
EXECUTING THE THIRD SET OF INSTRUCTIONS ON THE NETWORK UNTIL A PREDEFINED THRESHOLD IS REACHED, WHEREIN THE PREDEFINED THRESHOLDPERTAINS TO ACCURACY OF THE TRAINED NEURAL NETWORK IN THE REGION FOR THE ITERATION
312

FIG. 3


400
STOP TRAINING
IF ACCURACY > THRESHOLD_1 OR IF CHANGE IN ACCURACY < THRESHOLD_2 412
TRAIN DECISION TREE 414
GENERATE HYPERCUBE 416
SAMPLE HYPER CUBES 418
TRAIN ANN 408
TEST ANN 410
TESTING DATA 404
TRAINING DATA 406
SAMPLE UNIFORMLY ON THE INPUT SPACE 402

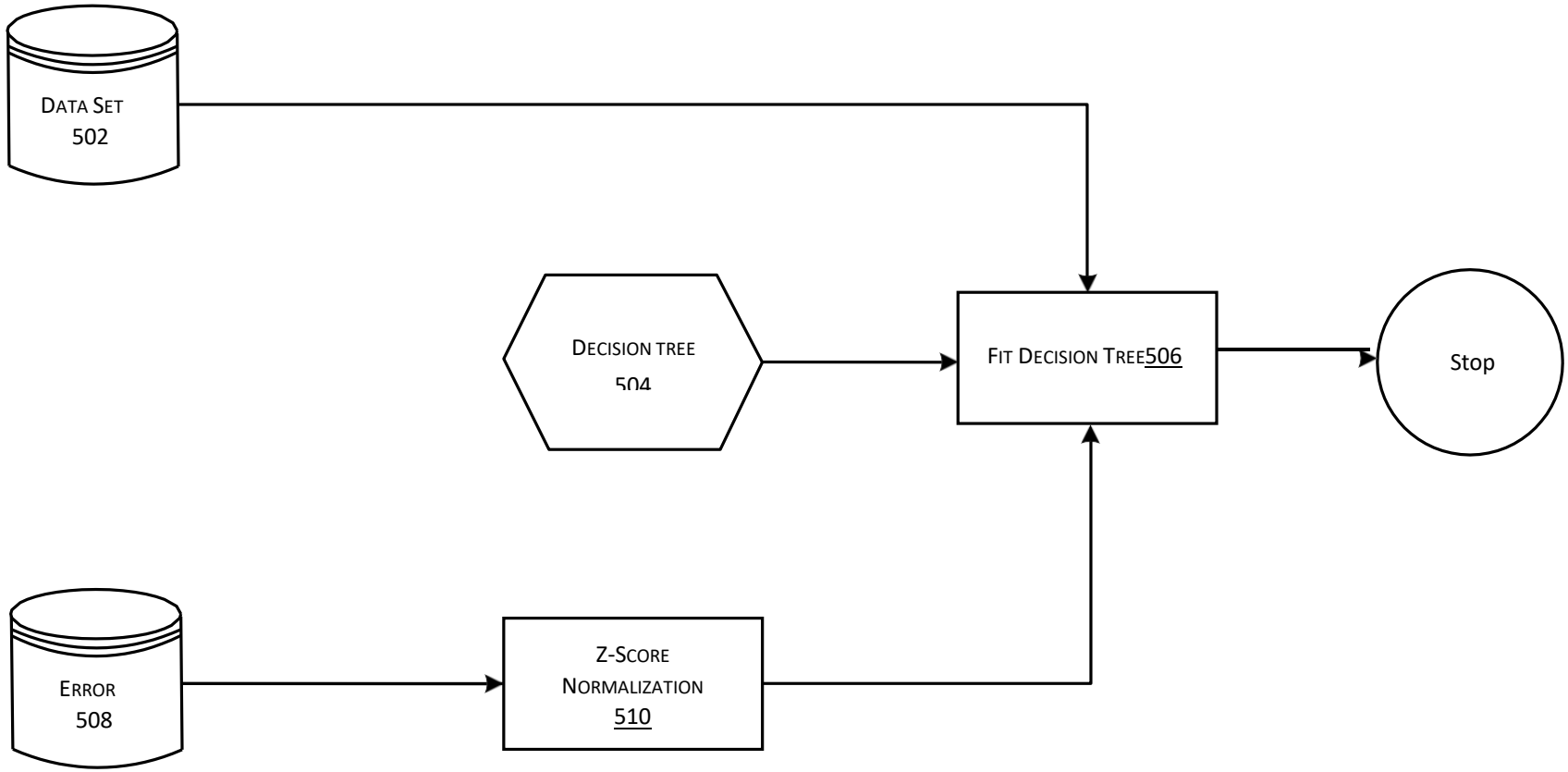

DATA SET
502
DECISION TREE
504
FIT DECISION TREE506
Stop
ERROR
508
Z-SCORE
NORMALIZATION
510

FIG. 5A

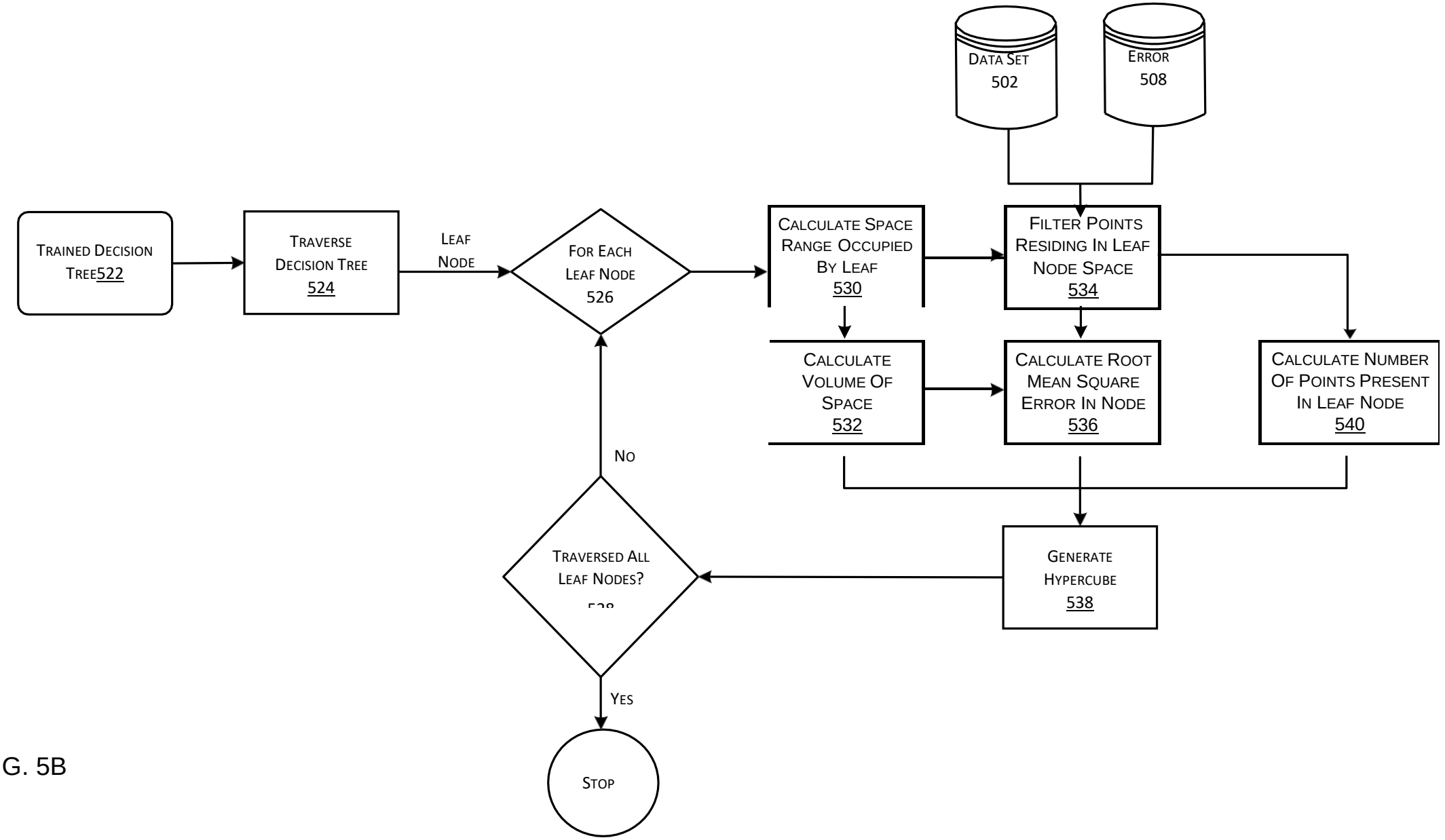

Trained Decision Tree522
Traverse Decision Tree 524
Leaf Node
For Each Leaf Node 526
Data Set 502
Error 508
Calculate Space Range Occupied By Leaf 530
Filter Points Residing In Leaf Node Space 534
Calculate Volume Of Space 532
Calculate Root Mean Square Error In Node 536
Calculate Number Of Points Present In Leaf Node 540
Generate Hypercube 538
Traversed All Leaf Nodes? 528
No
Yes
Stop

FIG. 5B

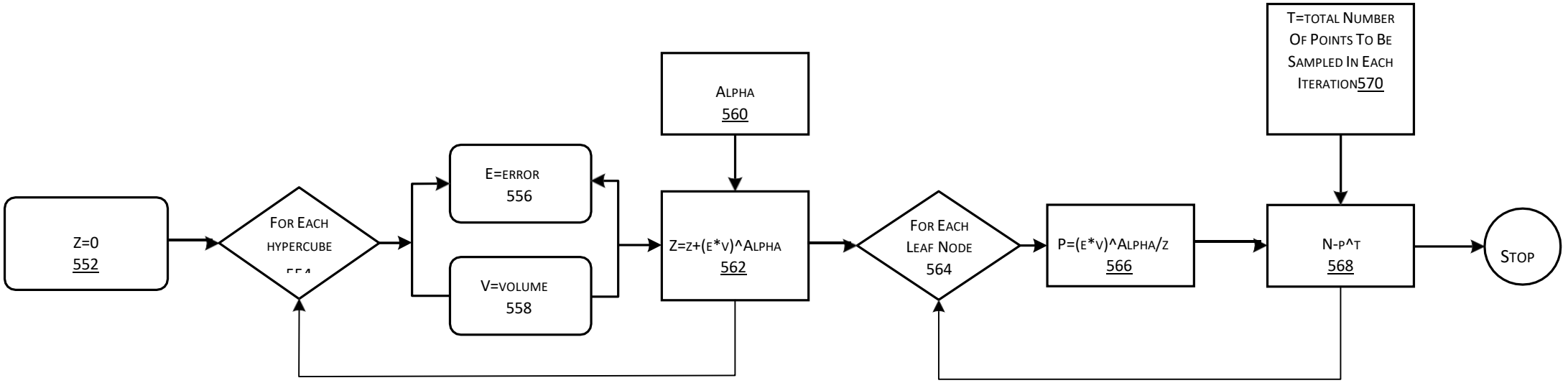

Z=0
552
For Each
hypercube
554
E=error
556
V=volume
558
Alpha
560
Z=z+(e*v)^Alpha
562
For Each
Leaf Node
564
P=(e*v)^Alpha/z
566
T=total Number
Of Points To Be
Sampled In Each
Iteration570
N-p^T
568
Stop

FIG. 5C

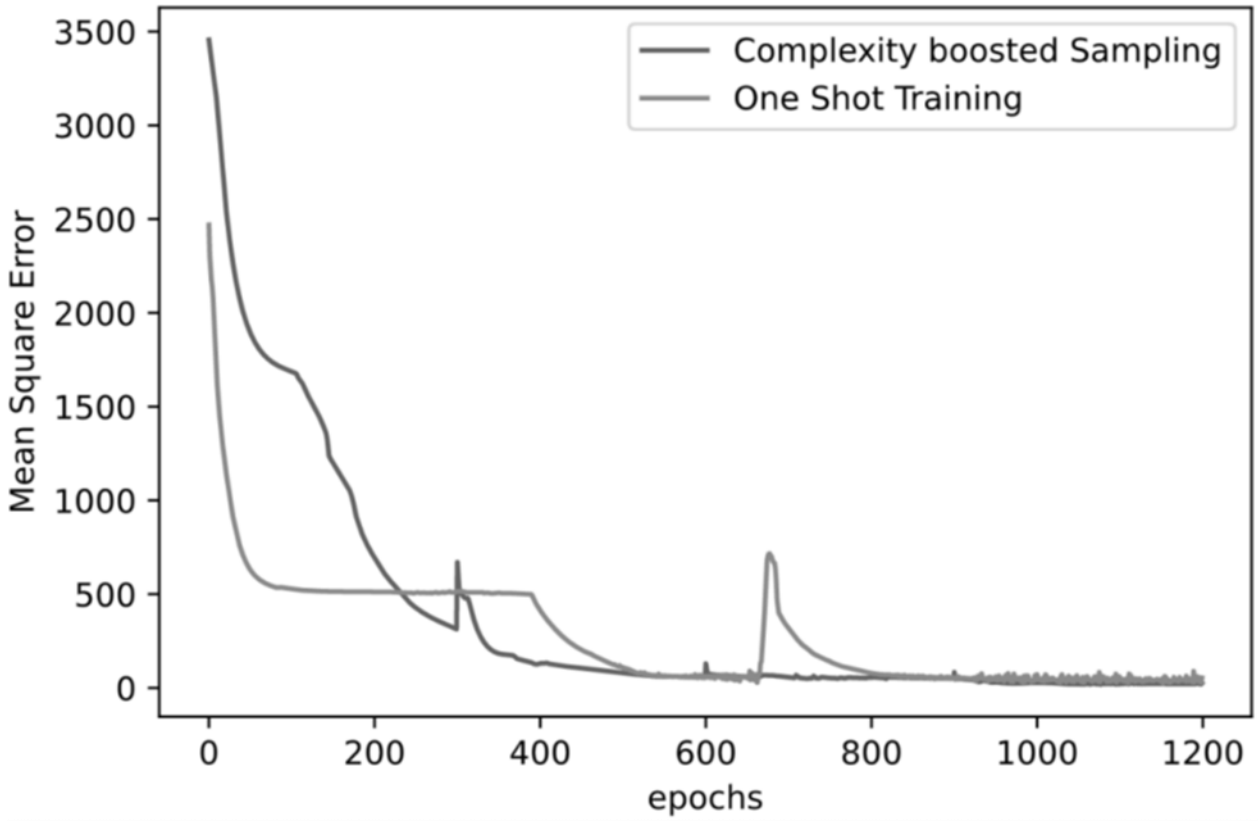

Complexity boosted Sampling
One Shot Training
Mean Square Error
3500
3000
2500
2000
1500
1000
500
0
0
200
400
600
800
1000
1200
epochs

FIG. 6D

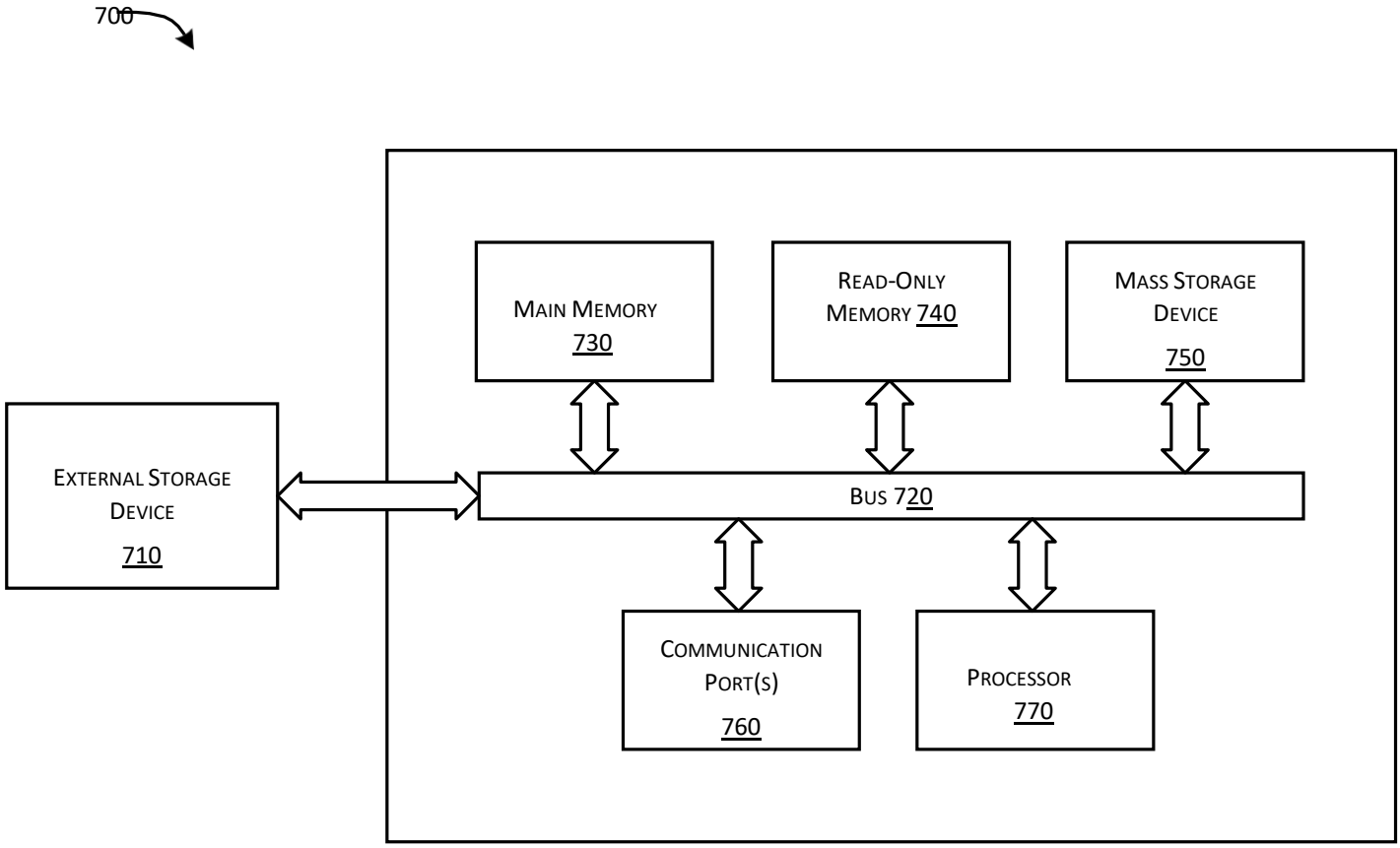

700
Main Memory
730
Read-Only
Memory 740
Mass Storage
Device
750
External Storage
Device
710
Bus 720
Communication
Port(s)
760
Processor
770

FIG. 7

FORM 2

THE PATENTS ACT, 1970

(39 OF 1970)

AND

THE PATENT RULES, 2003

PROVISIONAL SPECIFICATION

(See section 10 and rule 13)

SYSTEM AND METHOD FOR OPTIMIZING NON-LINEAR CONSTRAINTS OF AN INDUSTRIAL PROCESS UNIT

We, **JIO PLATFORMS LIMITED**, an Indian Citizen of, Office 101, Saffron, Nr. Centre Point, Panchwati 5 Rasta, Ambawadi, Ahmedabad-380006, Gujarat, India.

The following specification describes the invention:

ABSTRACT

SYSTEM AND METHOD FOR OPTIMIZING NON-LINEAR CONSTRAINTS OF AN INDUSTRIAL PROCESS UNIT

The present invention provides a robust and effective solution to an entity or an organization by enabling them to implement a system for facilitating creation of a digital twin of a process unit which can perform constrained optimization of control parameters to minimize or maximize an objective function. The system can capture non-linearities of the industrial process while the current Industrial Process models try to approximate non-linear process using linear approximation, which are not as accurate as Neural Networks. The proposed system can further create an end-to-end differentiable digital twin model of a process unit, and uses gradient flows for optimization as compared to other digital twin models that are gradient-free.

SYSTEM AND METHOD FOR OPTIMIZING NON-LINEAR CONSTRAINTS OF AN INDUSTRIAL PROCESS UNIT

FIELD OF INVENTION

[0059] The embodiments of the present disclosure generally relate to industrial process optimization using neural networks. More particularly, the present disclosure relates to a system and method for facilitating optimization of nonlinearities associated with an industrial process unit.

BACKGROUND OF THE INVENTION

[0060] The following description of related art is intended to provide background information pertaining to the field of the disclosure. This section may include certain aspects of the art that may be related to various features of the present disclosure. However, it should be appreciated that this section be used only to enhance the understanding of the reader with respect to the present disclosure, and not as admissions of prior art.

[0061] Adoption and implementation of automation technology is rapidly growing in several verticals of industry including Manufacturing, Refineries, Warehousing, Telecom etc. Any automation framework defines (1) Each process-unit and its input-to-output relationship, (2) How the process units interact with each

other for the functioning of the end-to-end process. It should also allow us to tune certain control-parameter inputs within each process unit so that a certain business objective for productivity/profitability etc. is met.

[0062] The current state of the art methodologies employs neural networks for performing non-linear control parameter optimization using various linear and non-linear optimization methods such as genetic search algorithms. Linear optimization for non-linear objective function gives accurate results only in a certain range while the non-linear methods only use the neural network in forward mode, that is, it only uses neural networks for calculating the outputs from the inputs. And the control parameters are changed further for optimization using various heuristics such as genetic search algorithms.

[0063] There is, therefore, a need in the art to provide a system and a method that enable improved ways of optimising non-linear process units, which are currently only being optimised using gradient-free linear methods which do not capture non linearities.

OBJECTS OF THE PRESENT DISCLOSURE

[0001] Some of the objects of the present disclosure, which at least one embodiment herein satisfies are as listed herein below.

[0064] It is an object of the present disclosure to provide a system and a method for facilitating a two stage ML based framework to perform constrained optimization on non-linear objective functions to find the optimal control-parameters for a given process-unit.

[0065] It is an object of the present disclosure to provide an approach that accurately captures its input-to-output relationship.

[0066] It is an object of the present disclosure to provide method that optimizes the control-parameter inputs towards a defined profit based objective function.

[0067] It is an object of the present disclosure to provide a system that enables creation of a digital twin of a process unit which can perform constrained optimization of control parameters to minimize or maximize an objective function.

[0068] It is an object of the present disclosure to provide a system that can capture non-linearities of the industrial process.

[0069] It is an object of the present disclosure to provide a system and method that creates an end-to-end differentiable digital twin model of a process unit, and uses gradient flows for optimization as compared to other digital twin models that are gradient-free.

BRIEF DESCRIPTION OF DRAWINGS

[0070] The accompanying drawings, which are incorporated herein, and constitute a part of this invention, illustrate exemplary embodiments of the disclosed methods and systems in which like reference numerals refer to the same parts throughout the different drawings. Components in the drawings are not necessarily to scale, emphasis instead being placed upon clearly illustrating the principles of the present invention. Some drawings may indicate the components using block diagrams and may not represent the internal circuitry of each component. It will be appreciated by those skilled in the art that invention of such drawings includes the invention of electrical components, electronic components or circuitry commonly used to implement such components.

[0071] FIG. 1 illustrates an exemplary network architecture in which or with which the system of the present disclosure can be implemented, in accordance with an embodiment of the present disclosure.

[0072] FIG. 2 illustrates an exemplary representation (200) of system (110) or a centralized server (112), in accordance with an embodiment of the present disclosure.

[0073] FIG. 3 illustrates exemplary block diagram representation depicting a Process unit, in accordance with an embodiment of the present disclosure.

[0074] FIG. 4 illustrates an exemplary representation of a two phase optimization architecture and its implementation, in accordance with an embodiment of the present disclosure.

[0075] FIG. 5 illustrates an exemplary representation (500) of a flowchart of accurately training a neural network with synthetically generated data using active complexity-based sampling, in accordance with an embodiment of the present disclosure.

[0076] FIGs. 6A-6B illustrate exemplary representation of at least two constraint functions, in accordance with an embodiment of the present disclosure.

[0077] FIG. 7 illustrates an exemplary representation (700) of a workflow of a process control parameter optimization system and its implementation, in accordance with an embodiment of the present disclosure.

[0078] FIGs. 8A-8C illustrate exemplary representations of results and analysis associated with the proposed method and system, in accordance with an embodiment of the present disclosure.

[0079] FIG. 9 illustrates an exemplary representation (900) of the flow process for secured access to decryption through the proposed big data analytical system, in accordance with an embodiment of the present disclosure.

[0080] FIG. 9 illustrates an exemplary computer system in which or with which embodiments of the present invention can be utilized in accordance with embodiments of the present disclosure.

[0081] The foregoing shall be more apparent from the following more detailed description of the invention.

BRIEF DESCRIPTION OF INVENTION

[0082] In the following description, for the purposes of explanation, various specific details are set forth in order to provide a thorough understanding of embodiments of the present disclosure. It will be apparent, however, that embodiments of the present disclosure may be practiced without these specific details. Several features described hereafter can each be used independently of one another or with any combination of other features. An individual feature may not address all of the problems discussed above or might address only some of the problems discussed above. Some of the problems discussed above might not be fully addressed by any of the features described herein.

[0083] The present invention provides a robust and effective solution to an entity or an organization by enabling them to implement a system for facilitating creation of a digital twin of a process unit which can perform constrained optimization of control parameters to minimize or maximize an objective function. The system can capture non-linearities of the industrial process while the current Industrial Process models try to approximate non-linear process using linear approximation, which are not as accurate as Neural Networks. The proposed system can further create an end-to-end differentiable digital twin model of a process unit, and uses gradient flows for optimization as compared to other digital twin models that are gradient-free.

[0084] Referring to FIG. 1 that illustrates an exemplary network architecture (100) in which or with which system (110) of the present disclosure can be implemented, in accordance with an embodiment of the present disclosure. As illustrated in FIG. 1, by way of example but not limitation, the exemplary architecture (100) may include a user (102) associated with a computing device (104), at least a network (106) and at least a centralized server (112). More specifically, the exemplary architecture (100) includes a system (110) equipped with a machine learning (ML) engine (216) for facilitating constrained optimization on non-linear attributes to find the optimal control-parameters for an industrial plant (120) (interchangeably referred to as industrial system or industrial process or industrial machine 120). The system (110) may be configured to receive a set of input signals to be transmitted to the industrial process (120). In an exemplary embodiment, the set of input signals may include any finite constant parameters or control parameters associated with an industrial process. In a way of example and not as a limitation, the constants and the control parameters may be concatenated to produce an input vector which can be fed into the process unit to generate output vector. The constants and controls are contextual and depend on the process. An example of a process unit is a

chemical unit in a refinery system. In a chemical unit, the set of control variables can be temperature, pressure, and feed flow, while the constants can be the composition of the input feed.

[0085] The system (110) may further be configured to train the set of inputs received by a causality learning module based on a predefined dataset to obtain the trained model. The predefined dataset may pertain to a dataset that may be synthetically generated by but limited to a forward mapping the constant parameters and control parameters to the output.

[0086] The system (110) may then be configured to optimize the trained model to obtain the accurate output signal.

[0087] In an exemplary embodiment, during the process of optimization, only the control parameters (hereinafter interchangeably referred to as variables) may be changed while the constant parameters may be kept unchanged using but not limited to back propagation. For example, in this system, temperature, pressure and feed flow can be changed iteratively in a constrained manner to optimize and objective function such as profit.

[0088] In an exemplary embodiment, the causality learning module (214) may be equipped with one or more neural networks that may be trained on the synthetically generated dataset that may be a forward mapping that maps constant parameters and control parameters to the output accurately. In an exemplary embodiment, the one or more neural networks can be trained in at least two ways such as but not limited to using pre-collected dataset and by sampling mathematical models which can simulate physical systems.

[0089] In an exemplary embodiment, a process objective function may be created which may capture a linear and boundary constraints of the control parameters of the industrial process (120). The control signals may be iteratively optimised using but not limited to Adam Optimiser by using backpropagated error. The backpropagation error may start at an output layer and gradient of the process objective function may be calculated with respect to the control parameters. For example, the Adam optimiser may use the gradient values to change the control parameters to minimise or maximise the process objective function.

[0090] In an exemplary embodiment, the system may be configured to stop the change in the control parameters when the process objective function is optimised

[0091] The centralised server (112) may include a database (210) that may store a knowledgebase having a set of potential parameters or information associated with the industrial process. The computing device (104) may be operatively coupled to the centralised server (112) through the network (106). In an exemplary embodiment, the knowledge base may be in the form of a hive table but not limited to the like.

[0092] In an embodiment, the system (110)/ server (112) may further configure the ML engine (216) to generate, through an appropriately selected machine learning (ML) model of the system in a way of example

and not as limitation, a trained model configured to process and optimize the set of input signal received, and predict to read actual parameters associated with the optimized industrial process. The trained model may enable lookup with the faster storage database to get the optimized parameters and add it to the existing dataset as a new field and write it into a destination dataset.

[0093] In an embodiment, the computing device (104) may communicate with the system (110) via set of executable instructions residing on any operating system, including but not limited to, Android TM, iOS TM, Kai OS TM and the like. In an embodiment, computing device (104) may include, but not limited to, any electrical, electronic, electro-mechanical or an equipment or a combination of one or more of the above devices such as mobile phone, smartphone, virtual reality (VR) devices, augmented reality (AR) devices, laptop, a general-purpose computer, desktop, personal digital assistant, tablet computer, mainframe computer, or any other computing device, wherein the computing device may include one or more in-built or externally coupled accessories including, but not limited to, a visual aid device such as camera, audio aid, a microphone, a keyboard, input devices for receiving input from a user such as touch pad, touch enabled screen, electronic pen and the like. It may be appreciated that the computing device (104) may not be restricted to the mentioned devices and various other devices may be used. A smart computing device may be one of the appropriate systems for storing data and other private/sensitive information.

[0094] In an exemplary embodiment, a network (106) may include, by way of example but not limitation, at least a portion of one or more networks having one or more nodes that transmit, receive, forward, generate, buffer, store, route, switch, process, or a combination thereof, etc. one or more messages, packets, signals, waves, voltage or current levels, some combination thereof, or so forth. A network may include, by way of example but not limitation, one or more of: a wireless network, a wired network, an internet, an intranet, a public network, a private network, a packet-switched network, a circuit-switched network, an ad hoc network, an infrastructure network, a public-switched telephone network (PSTN), a cable network, a cellular network, a satellite network, a fiber optic network, some combination thereof.

[0095] In another exemplary embodiment, the centralized server (112) may include or comprise, by way of example but not limitation, one or more of: a stand-alone server, a server blade, a server rack, a bank of servers, a server farm, hardware supporting a part of a cloud service or system, a home server, hardware running a virtualized server, one or more processors executing code to function as a server, one or more machines performing server-side functionality as described herein, at least a portion of any of the above, some combination thereof.

[0096] In an embodiment, the system (110) may include one or more processors coupled with a memory, wherein the memory may store instructions which when executed by the one or more processors

may cause the system to perform the generation of automated visual responses to a query. FIG. 2 with reference to FIG. 1, illustrates an exemplary representation of system (110) /centralized server (112) for facilitating constrained optimization on non-linear attributes of an industrial process (120) to find the optimal control-parameters for a given process-unit based on a machine learning based architecture, in accordance with an embodiment of the present disclosure. In an aspect, the system (110) /centralized server (112) may comprise one or more processor(s) (202). The one or more processor(s) (202) may be implemented as one or more microprocessors, microcomputers, microcontrollers, digital signal processors, central processing units, logic circuitries, and/or any devices that process data based on operational instructions. Among other capabilities, the one or more processor(s) (202) may be configured to fetch and execute computer-readable instructions stored in a memory (206) of the system (110). The memory (206) may be configured to store one or more computer-readable instructions or routines in a non-transitory computer readable storage medium, which may be fetched and executed to create or share data packets over a network service. The memory (206) may comprise any non-transitory storage device including, for example, volatile memory such as RAM, or non-volatile memory such as EPROM, flash memory, and the like.

[0097] In an embodiment, the system (110)/centralized server (112) may include an interface(s) 204. The interface(s) 204 may comprise a variety of interfaces, for example, interfaces for data input and output devices, referred to as I/O devices, storage devices, and the like. The interface(s) 204 may facilitate communication of the system (110). The interface(s) 204 may also provide a communication pathway for one or more components of the system (110) or the centralized server (112). Examples of such components include, but are not limited to, processing engine(s) 208 and a database 210.

[0098] The processing engine(s) (208) may be implemented as a combination of hardware and programming (for example, programmable instructions) to implement one or more functionalities of the processing engine(s) (208). In examples described herein, such combinations of hardware and programming may be implemented in several different ways. For example, the programming for the processing engine(s) (208) may be processor executable instructions stored on a non-transitory machine-readable storage medium and the hardware for the processing engine(s) (208) may comprise a processing resource (for example, one or more processors), to execute such instructions. In the present examples, the machine-readable storage medium may store instructions that, when executed by the processing resource, implement the processing engine(s) (208). In such examples, the system (110) /centralized server (112) may comprise the machine-readable storage medium storing the instructions and the processing resource to execute the instructions, or the machine-readable storage medium may be separate but accessible to the system (110) /centralized server (112)

and the processing resource. In other examples, the processing engine(s) (208) may be implemented by electronic circuitry.

[0099] The processing engine (208) may include one or more engines selected from any of a signal acquisition engine (212), a causality learning module or engine (214), a machine learning (ML) engine (216), a trained model generation engine (218) and other engines (220).

[00100] FIG. 3 illustrates exemplary block diagram representation depicting a Process unit (300), in accordance with an embodiment of the present disclosure. As illustrated, in an aspect the process unit (300) may include parameters where,

X is the constant vector

u is the control vector

y is the output vector and is a function of constant vector X and control vector u

i is the iteration during the process of optimization

X and *u* are concatenated and fed into the process unit to generate y

[00101] FIG. 4 illustrates an exemplary representation of a two phase optimization architecture and its implementation, in accordance with an embodiment of the present disclosure. As illustrated, in an aspect, the process of optimization of the process unit may include at least a two-step process: such as Causality Learning (402) and Control Parameters Optimization (404) using backpropagation. In the phase of causality learning (402), a neural network is trained on a synthetically generated dataset i.e. a forward mapping is learnt that maps constant parameters and control parameters to the output accurately. The Neural Networks can be trained in two ways: Using pre-collected dataset and by sampling mathematical models which can simulate physical systems.

[00102] In an exemplary embodiment, the system may generate synthetic data by adaptive sampling. If the mathematical model of the physical process is not completely accurate, the synthetic data can be combined with process data from the physical process to generate a hybrid dataset. This hybrid dataset can be further used to fine tune the trained neural network.

[00103] FIG. 5 illustrates an exemplary representation (500) of a flowchart of accurately training a neural network with synthetically generated data using active complexity-based sampling, in accordance with an embodiment of the present disclosure.

[00104] As illustrated, in an aspect, once the causality is learned by the Neural Network (NN), it is used to optimize controls for an objective function. The Objective function (504) can have multiple constraints for the controls, like, boundary constraints and linear constraints (502). Then synthetic data may be generated by adaptive sampling or active complexity based sampling (506) to obtain an unlimited data across the input

output space (508) that may be the data required for training an ANN (510). The Neural Networks (512 and 514) can be used to perform optimization by gradient based methods. Modern NN programming uses dynamic computational graphs for maintaining a record of gradients for the parameters of NN. These gradients can be used to perform gradient descent on the controls for optimizing an objective function. Multiple smaller objective functions can be combined together using Lagrange Multipliers to create a global objective function. These Lagrange Multipliers will act as hyper-parameters for the optimization pipeline.

[00105] Following is an example of a generic objective function used to maximize profits while being limited by boundary and linear constraints on the controls side. Given a Product Revenue ***R***, Input Cost ***S*** and Operation Cost ***O*** for a Digital Twin, Profit function **J** can be defined as

$$J(f,c) = -(R(f,c) - S(f) - O(c)) - \lambda_{c1}L_{c1}(c) - \lambda_{F1}L_{F1}(f) - \lambda_{c2}L_{c2}(c) - \lambda_{F2}L_{F2}(f)$$

where,

f, ***c*** are process control parameters
λ_{c1}, λ_{c2}, λ_{F1} and λ_{F2} are Lagrange multipliers
L_{c1}, L_{c2} , L_{F1} and L_{F2} are constraint functions for ***f*** and ***c*** respectively

[00106] FIGs. 6A-6B illustrate exemplary representation of at least two constraint functions, in accordance with an embodiment of the present disclosure.

[00107] FIG. 6A shows an exemplary Boundary Constraint function where boundary constraints (L_{c1}, L_{F1}) function may provide a min-max range for the process controls to be optimised in, and are defined by

$$\underset{F1}{L_{c1}}(\bar{c}) = \sum_k \sigma_\lambda(C_k^{min} - c_k, \alpha) + \sigma_\lambda(c_k - C_k^{max}, \alpha)$$

- k is the number of variables in the control vector
- C_k^{min} and C_k^{max} are the min-max values for the variable c_k
- $\sigma_\lambda(x, \alpha)$ is a modified sigmoid function and is defined as
- $\sigma_\lambda(x, \alpha) = \frac{1}{1+e^{-\alpha * x}}$
- α is a hyperparameter whose value is at least 500 but not limited to it.

[00108] FIG. 6B illustrates a linear constraint function constraints the process control variables to follow a linear constraint and is defined as

$$\underset{F2}{L_{c2}} = \sum_k RMSE(Q - \theta_k c_k)$$

where:

- $RMSE$ is the root mean square error function

- Q is a constant
- θ_k is the coefficient of c_k

[00109] FIG. 7 illustrates an exemplary representation (700) of a workflow of a process control parameter optimization system and its implementation, in accordance with an embodiment of the present disclosure. As illustrated, the Process Control Parameter Optimization may be performed once the objective function may be formulated and includes the constraint functions such as the constants (702) and control parameters (704) that may be concatenated (706) and passed on to a neural network (708) to obtain the required output (710) from which an objective function is calculated (712) which may be the Loss to be backpropagated (714) is the objective function.

[00110] For example, given constraints can be

- PRICE of all outputs
- COST of all inputs
- Trained Neural Network
- Initial value of input $\bar{\boldsymbol{f}}_0$ and control $\bar{\boldsymbol{c}}_0$ parameters

[00111] In an exemplary embodiment, optimization of feed and control using gradient ascent may include the following steps

- Do a forward pass for the current $\bar{\boldsymbol{f}}_t$ and $\bar{\boldsymbol{c}}_t$ and estimate profit $=$ Profit$(\bar{\boldsymbol{f}}_t, \bar{\boldsymbol{c}}_t)$
- The objective function is calculated
- Loss to be backpropagated is the objective function
- This error DOES NOT update any of the weights in the already trained neural network
- This error propagates ALL THE WAY to the control vectors
- We update the control vectors based on this backpropagated error using Adam Optimizer
- If maximum percentage change in control/input parameter is < threshold, then stop
- Otherwise go to step 1

[00112] FIGs. 8A-8C illustrate exemplary representations of results and analysis associated with the proposed method and system, in accordance with an embodiment of the present disclosure. FIG. 8A, FIG. 8B and FIG. 8C show that profitability is maximised ensuring boundary and linear constraints are not violated and estimated optimized profit is highly accurate.

[00113] FIG. 9 illustrates an exemplary computer system in which or with which embodiments of the present invention can be utilized in accordance with embodiments of the present disclosure. As shown in FIG. 9, computer system (900) can include an external storage device (910), a bus (920), a main memory (930), a read only memory (940), a mass storage device (970), communication port (960), and a processor (970). A

person skilled in the art will appreciate that the computer system may include more than one processor and communication ports. Examples of processor (970) include, but are not limited to, an Intel® Itanium® or Itanium 2 processor(s), or AMD® Opteron® or Athlon MP® processor(s), Motorola® lines of processors, FortiSOC™ system on chip processors or other future processors. Processor (970) may include various modules associated with embodiments of the present invention. Communication port (960 can be any of an RS-232 port for use with a modem based dialup connection, a 10/100 Ethernet port, a Gigabit or 9 Gigabit port using copper or fiber, a serial port, a parallel port, or other existing or future ports. Communication port (960) may be chosen depending on a network, such a Local Area Network (LAN), Wide Area Network (WAN), or any network to which computer system connects. Memory (930) can be Random Access Memory (RAM), or any other dynamic storage device commonly known in the art. Read-only memory (940) can be any static storage device(s) e.g., but not limited to, a Programmable Read Only Memory (PROM) chips for storing static information e.g., start-up or BIOS instructions for processor (970). Mass storage (950) may be any current or future mass storage solution, which can be used to store information and/or instructions. Exemplary mass storage solutions include, but are not limited to, Parallel Advanced Technology Attachment (PATA) or Serial Advanced Technology Attachment (SATA) hard disk drives or solid-state drives (internal or external, e.g., having Universal Serial Bus (USB) and/or Firewire interfaces), e.g. those available from Seagate (e.g., the Seagate Barracuda 792 family) or Hitachi (e.g., the Hitachi Deskstar 7K1000), one or more optical discs, Redundant Array of Independent Disks (RAID) storage, e.g. an array of disks (e.g., SATA arrays), available from various vendors including Dot Hill Systems Corp., LaCie, Nexsan Technologies, Inc. and Enhance Technology, Inc.

[00114] Bus (920) communicatively couples processor(s) (970) with the other memory, storage and communication blocks. Bus (920) can be, e.g. a Peripheral Component Interconnect (PCI) / PCI Extended (PCI-X) bus, Small Computer System Interface (SCSI), USB or the like, for connecting expansion cards, drives and other subsystems as well as other buses, such a front side bus (FSB), which connects processor (970) to software system.

[00115] Optionally, operator and administrative interfaces, e.g. a display, keyboard, joystick and a cursor control device, may also be coupled to bus (920) to support direct operator interaction with a computer system. Other operator and administrative interfaces can be provided through network connections connected through communication port (960). The external storage device (99) can be any kind of external hard-drives, floppy drives, IOMEGA® Zip Drives, Compact Disc - Read Only Memory (CD-ROM), Compact Disc-Re-Writable (CD-RW), Digital Video Disk-Read Only Memory (DVD-ROM). Components described above are

meant only to exemplify various possibilities. In no way should the aforementioned exemplary computer system limit the scope of the present disclosure.

[00116] Thus, the present disclosure provides a unique and inventive solution for optimising non-linear process units, which are currently only being optimized using gradient-free linear methods which do not capture non linearities. An optimization system / model using ANN helps to manage operations in its true nature of state, capturing dynamic interactions of all the measured variables in input and output space. Using such system in plant operations may provide maximum productivity and efficiency from its assets.

[00117] While considerable emphasis has been placed herein on the preferred embodiments, it will be appreciated that many embodiments can be made and that many changes can be made in the preferred embodiments without departing from the principles of the invention. These and other changes in the preferred embodiments of the invention will be apparent to those skilled in the art from the disclosure herein, whereby it is to be distinctly understood that the foregoing descriptive matter to be implemented merely as illustrative of the invention and not as limitation.

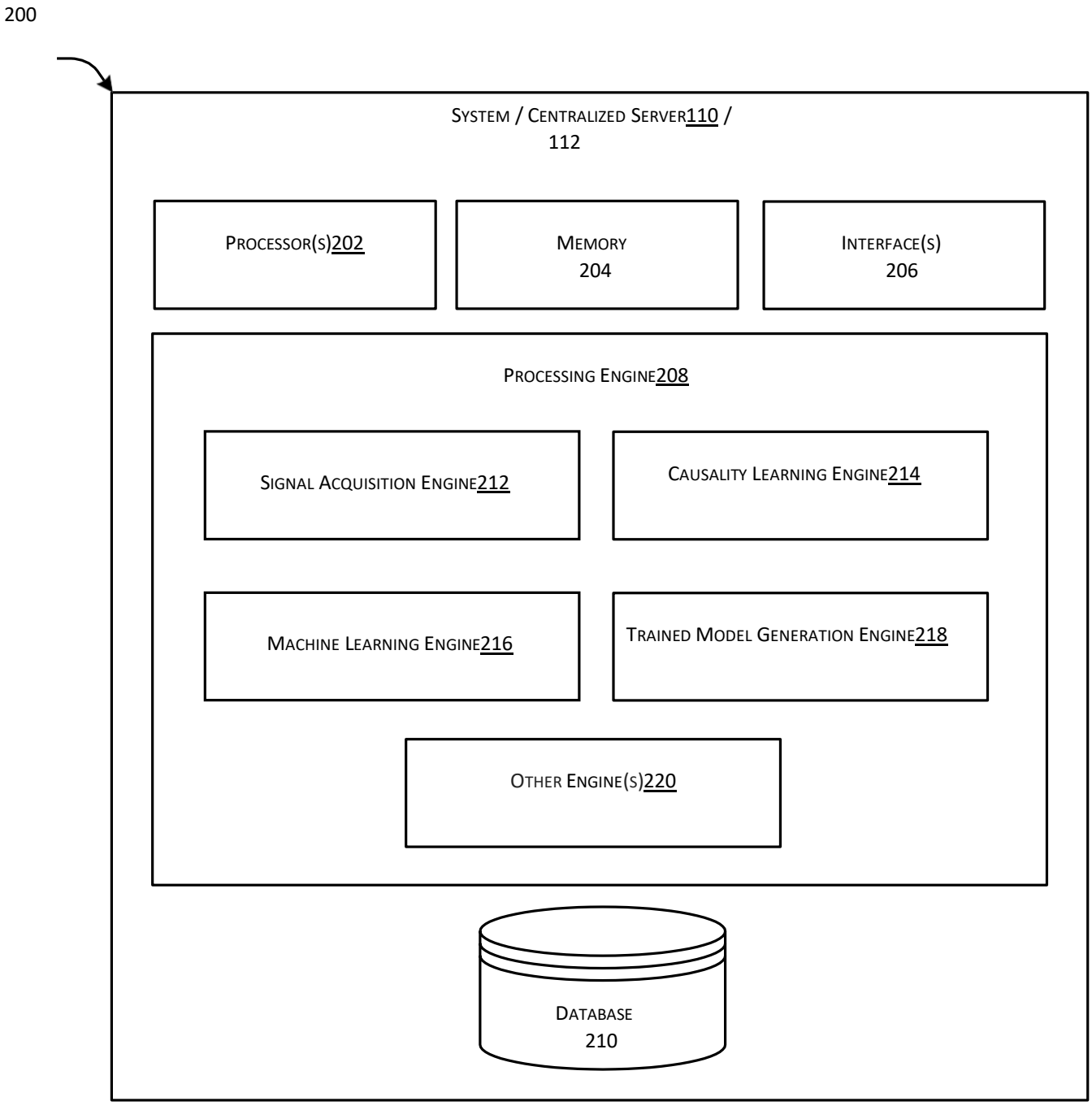

200
SYSTEM / CENTRALIZED SERVER110 /
112
PROCESSOR(S)202
MEMORY
204
INTERFACE(S)
206
PROCESSING ENGINE208
SIGNAL ACQUISITION ENGINE212
CAUSALITY LEARNING ENGINE214
MACHINE LEARNING ENGINE216
TRAINED MODEL GENERATION ENGINE218
OTHER ENGINE(S)220
DATABASE
210

FIG. 2

300

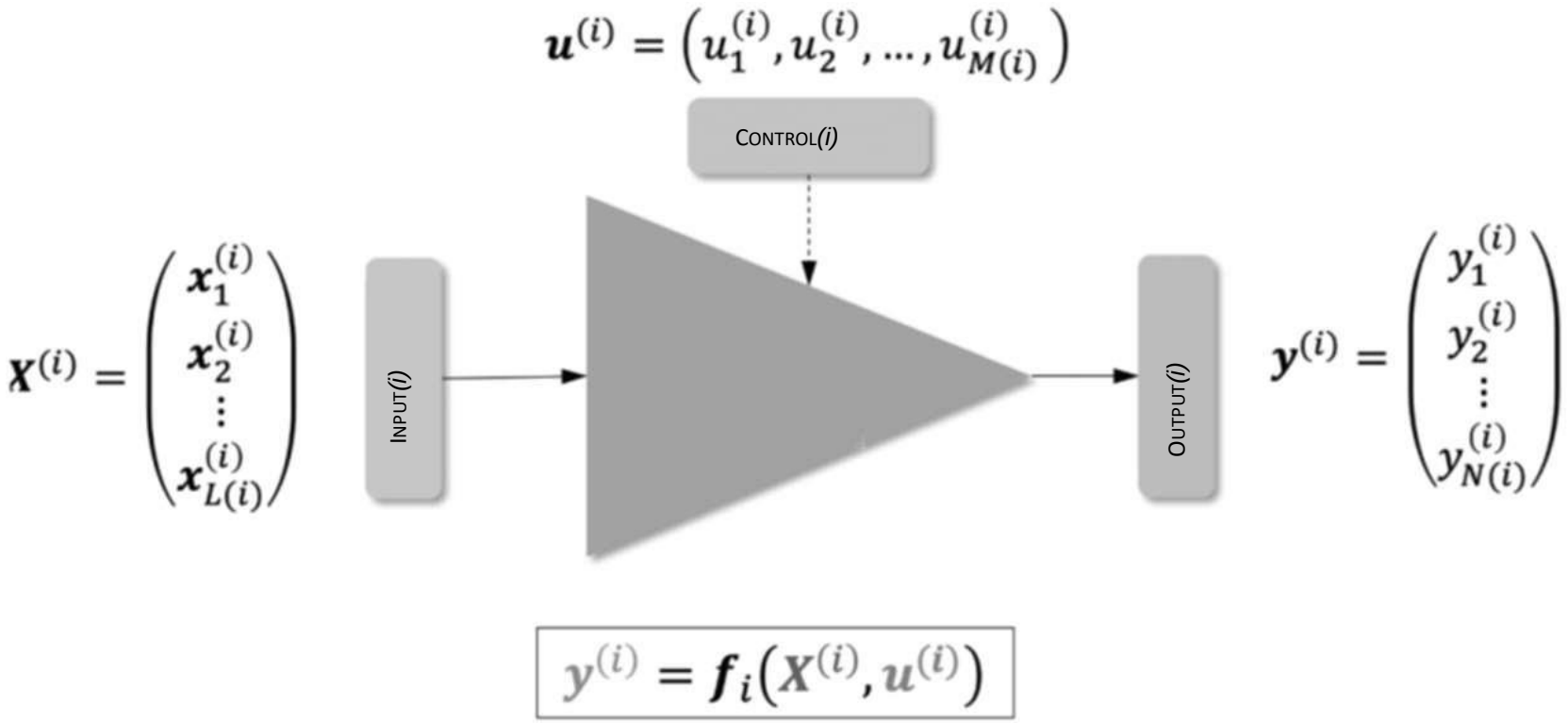
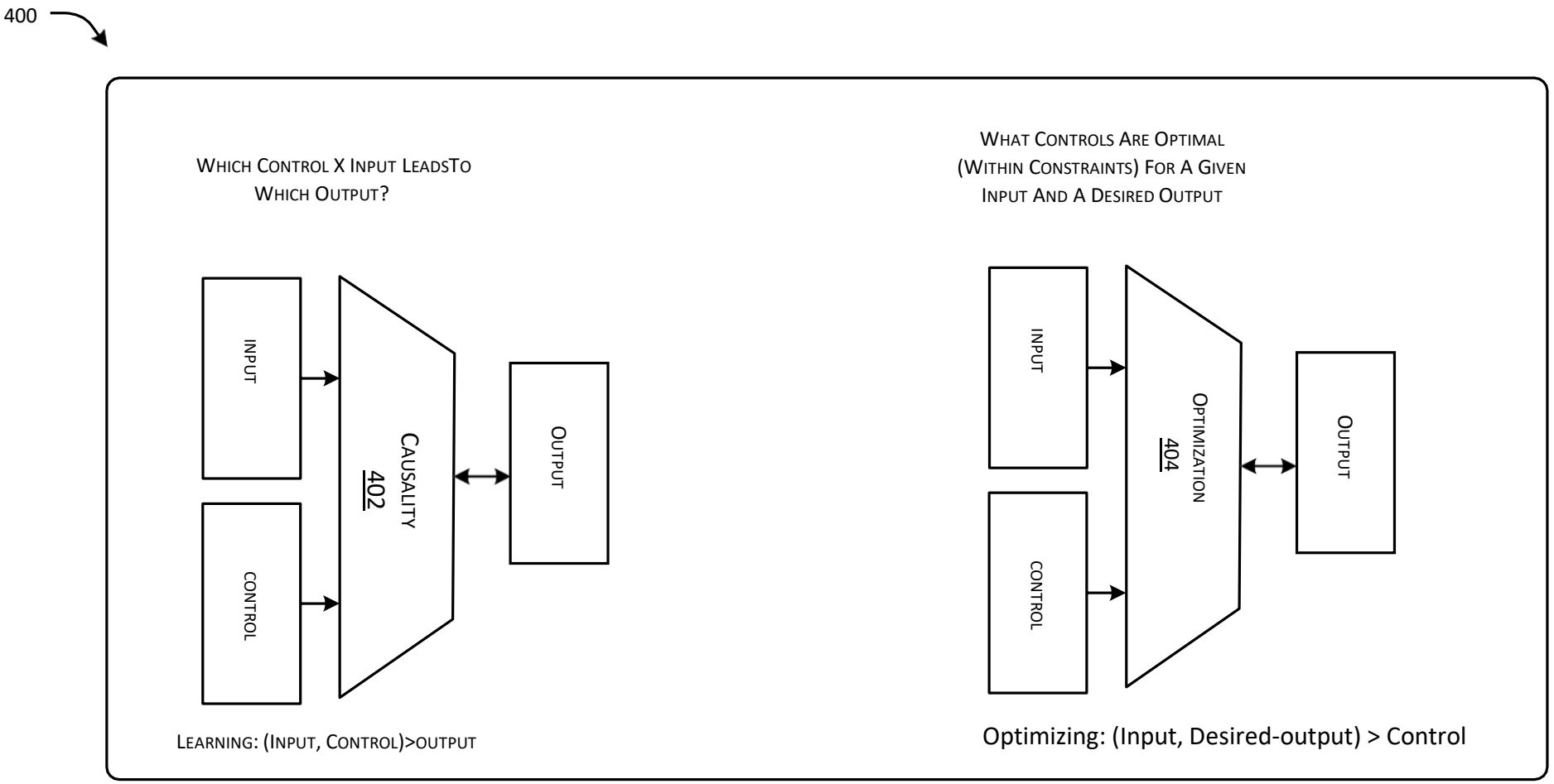

$u^{(i)} = \left(u_1^{(i)}, u_2^{(i)}, \ldots, u_{M(i)}^{(i)}\right)$
CONTROL(*i*)
$X^{(i)} = \begin{pmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_{L(i)}^{(i)} \end{pmatrix}$
INPUT(*i*)
OUTPUT(*i*)
$y^{(i)} = \begin{pmatrix} y_1^{(i)} \\ y_2^{(i)} \\ \vdots \\ y_{N(i)}^{(i)} \end{pmatrix}$
$y^{(i)} = f_i\left(X^{(i)}, u^{(i)}\right)$

FIG. 3


400
WHICH CONTROL X INPUT LEADSTO WHICH OUTPUT?
INPUT
CONTROL
CAUSALITY 402
OUTPUT
LEARNING: (INPUT, CONTROL)>OUTPUT
WHAT CONTROLS ARE OPTIMAL (WITHIN CONSTRAINTS) FOR A GIVEN INPUT AND A DESIRED OUTPUT
INPUT
CONTROL
OPTIMIZATION 404
OUTPUT
Optimizing: (Input, Desired-output) > Control

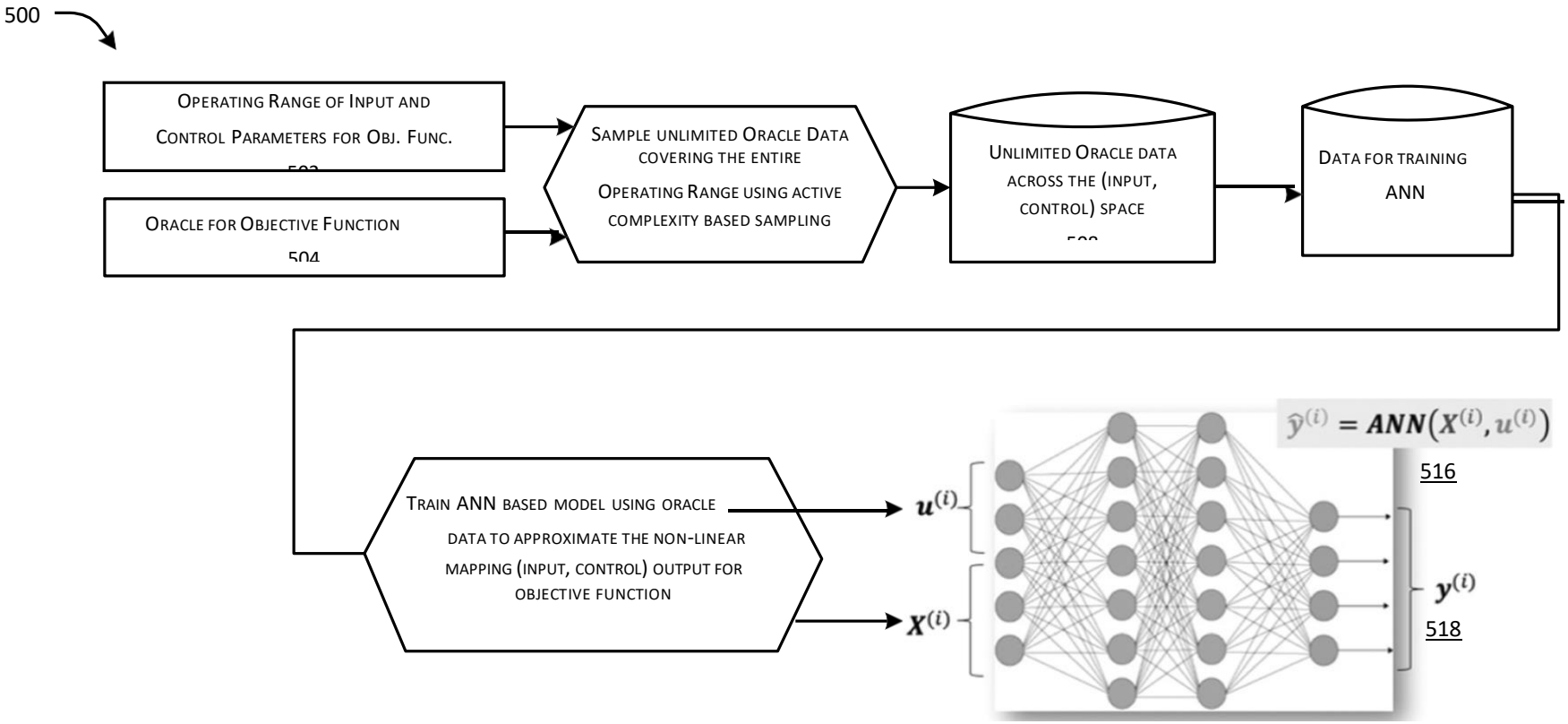

500
OPERATING RANGE OF INPUT AND CONTROL PARAMETERS FOR OBJ. FUNC.
502
ORACLE FOR OBJECTIVE FUNCTION
504
SAMPLE UNLIMITED ORACLE DATA COVERING THE ENTIRE OPERATING RANGE USING ACTIVE COMPLEXITY BASED SAMPLING
UNLIMITED ORACLE DATA ACROSS THE (INPUT, CONTROL) SPACE
508
DATA FOR TRAINING ANN
TRAIN ANN BASED MODEL USING ORACLE DATA TO APPROXIMATE THE NON-LINEAR MAPPING (INPUT, CONTROL) OUTPUT FOR OBJECTIVE FUNCTION
$u^{(i)}$
$X^{(i)}$
$\hat{y}^{(i)} = ANN(X^{(i)}, u^{(i)})$
516
$y^{(i)}$
518

FIG. 5

600

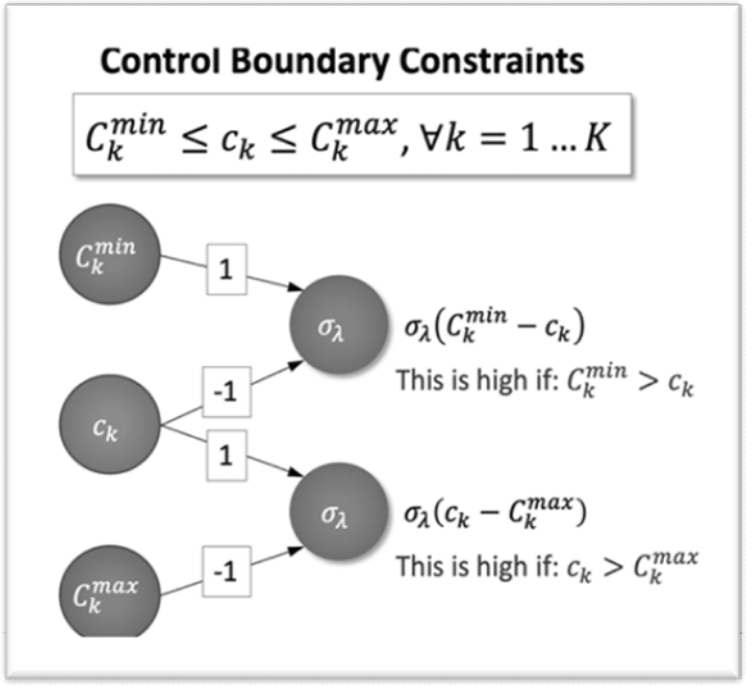


FIG. 6A

620

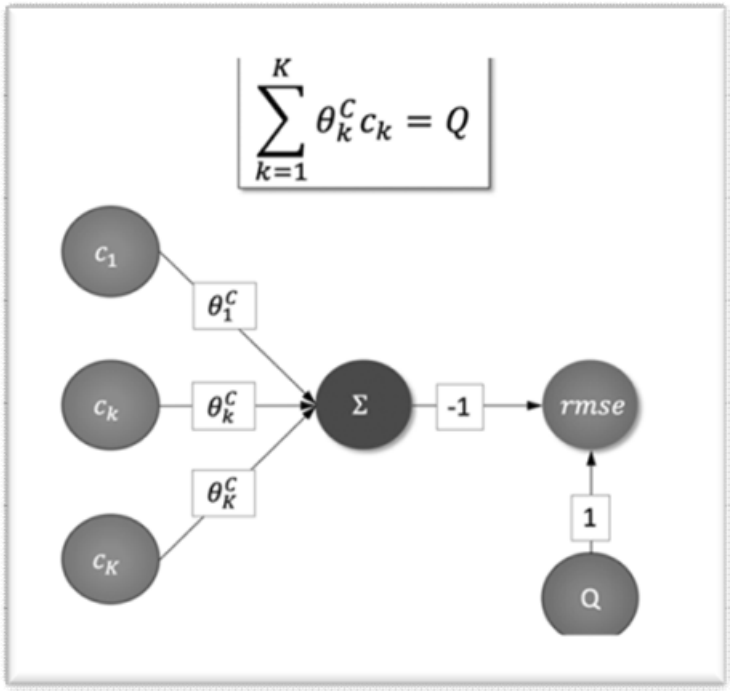


FIG. 6B

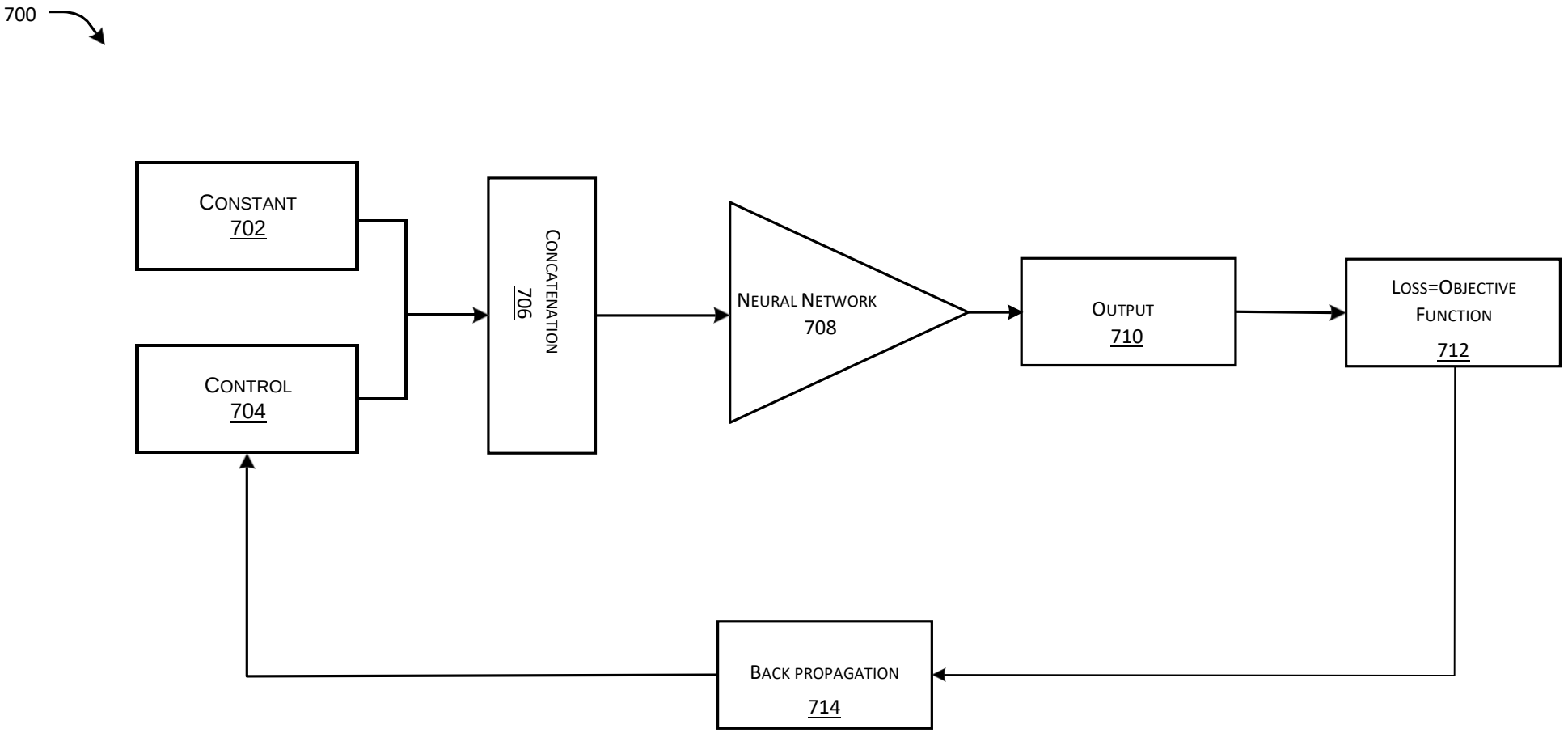

700
CONSTANT
702
CONTROL
704
CONCATENATION
706
NEURAL NETWORK
708
OUTPUT
710
LOSS=OBJECTIVE FUNCTION
712
BACK PROPAGATION
714

FIG. 7

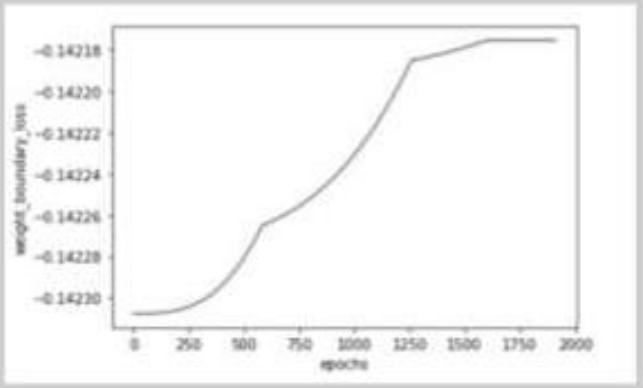
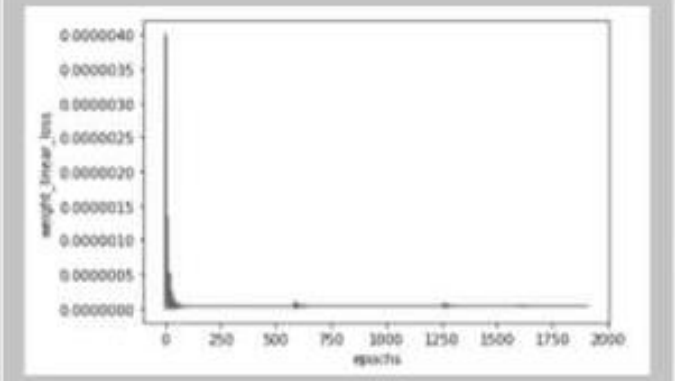
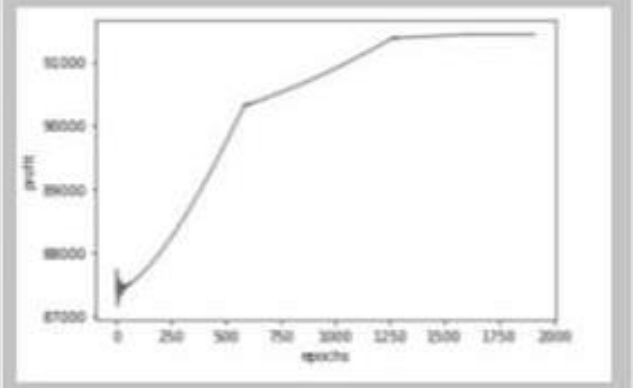
	BOUNDARY CONSTRAINTS	LINEAR CONSTRAINTS	PROFITABILITY
FEED			
	$F_n^{min} \leq f_n \leq F_n^{max}$ $\forall n = 1 \ldots N$	$F_{sum}^{min} \leq \sum_{n=1}^{N} \theta_n^F f_n \leq F_{sum}^{max}$	$\mathrm{R}(\bar{f}, \bar{c}) - \mathrm{S}(\bar{f})$

FIG. 8A FIG. 8B FIG. 8C

- ✓ Profitability is maximized ensuring boundary and linear constraints are not violated
- ✓ Estimated Optimized Profit is highly accurate

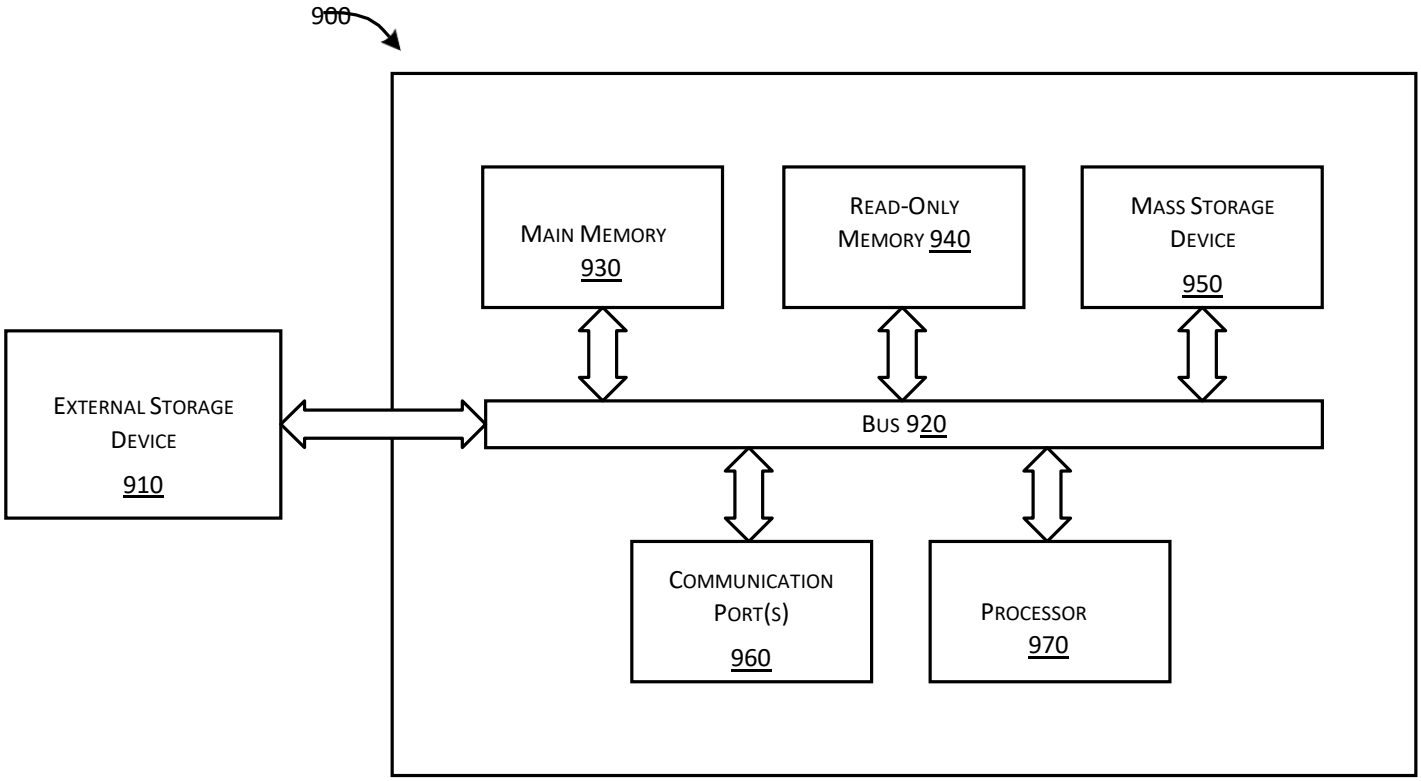

900
MAIN MEMORY
930
READ-ONLY
MEMORY 940
MASS STORAGE
DEVICE
950
EXTERNAL STORAGE
DEVICE
910
BUS 920
COMMUNICATION
PORT(S)
960
PROCESSOR
970

FIG. 9